SURFACE SEARCH RADAR TRACKING BY A MICROCOMPUTER KALMAN FILTER

Charles Howard Wilson

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SURFACE SEARCH RADAR TRACKING BY A MICROCOMPUTER KALMAN FILTER

by

Charles Howard Wilson

June 1976

Thesis Advisor: V. M. Powers

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Surface Search Radar Tracking by a Microcomputer Kalman Filter	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis; June 1976
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Charles Howard Wilson	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: June 1976 / 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Naval Postgraduate School, Monterey, California 93940	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprocessor, Microcomputer, Kalman Filter, Surface Search Radar, Radar Tracking

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

U. S. Navy ships not equipped with NTDS currently perform surface radar target tracking manually, a tedious and inaccurate process.

This thesis establishes that an 8-bit microprocessor, interfaced to a common surface search radar, could compute the course and speed of multiple radar targets. The investigation includes the estimation of radar measurement errors, a determination of practical digital interface limitations, and the development

of a Kalman filter algorithm, a floating point arithmetic library for the INTEL 8080 microprocessor, and the microprocessor tracking system software. The error statistics and execution time of the microprocessor software are presented for several computer simulated target tracks.

Surface Search Radar Tracking by a Microcomputer Kalman Filter

by

Charles Howard Wilson
Lieutenant, United States Navy
B.S., University of Idaho, 1970

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1976

ABSTRACT

U. S. Navy ships not equipped with NTDS currently perform surface radar target tracking manually, a tedious and inaccurate process.

This thesis establishes that an 8-bit microprocessor, interfaced to a common surface search radar, could compute the course and speed of multiple radar targets. The investigation includes the estimation of radar measurement errors, a determination of practical digital interface limitations, and the development of a Kalman filter algorithm, a floating point arithmetic library for the INTEL 8080 microprocessor, and the microprocessor tracking system software. The error statistics and execution time of the microprocessor software are presented for several computer simulated target tracks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

"In December 1973 INTEL Corporation shipped the first 8-bit N-channel microprocessor, the 8080." [1] In the months that followed, several semiconductor manufacturers introduced their own version of INTEL's 8080 microprocessor. The resulting competition has driven microprocessor prices down and performance up. Advanced Micro Devices announced in April 1976, their AM9080A-4 microprocessor, with an instruction execution time one-half that of INTEL's original 8080, was available for $21.00 [2]. In the same announcement, they predicted the AM9080A would cost $5.00 by 1980. Paralleling the advances in microprocessor technology, the manufacturers of semiconductor memories are making steady progress in expanding the memory capability of single semiconductor chips, reducing memory access time and subsequently reducing the cost of high-speed semiconductor memories used with microprocessors.

These advances in semiconductor technology are taking place at a time when the U. S. Navy is debating the cost of rebuilding the fleet. A significant portion of the cost of building a modern warship is the electronics in its weapons systems. Microprocessors offer the potential to (1) reduce the cost of digital systems, (2) perform complex functions at remote stations, relieving the congestion at larger central computing facilities, and (3) perform functions currently handled by watch personnel, thus reducing

the manning requirements of watch sections.

A. OBJECTIVES OF THESIS

The primary objective of this thesis is to demonstrate the capability of microprocessors to perform surface search radar tracking, thus relieving the tedium of this function from the typical underway watch section.

The secondary objective of this thesis is to investigate the performance of a Kalman filter in the two-dimensional polar coordinate system, with measurement noise, typical of a surface search radar.

B. SCOPE OF THE WORK

The scope of the work was limited to the development of a Kalman filter tracking algorithm for the INTEL 8080 microprocessor and simulation of circuitry necessary to interface with the AN/SPS-10 surface search radar.

A Kalman filter was chosen because it is computationally the most complex of the tracking filters typically used for radar tracking. Thus, it would be the most demanding on the microprocessor. Additionally, the Kalman filter is seldom implemented in its complete form due to its computational complexity thus adding additional relevance to this investigation.

The parameters for the AN/SPS-10 surface search radar were used for computations requiring specific radar parameters. This radar was selected because it was available at the Naval Postgraduate School and is installed aboard

the majority of U. S. Naval ships.

INTEL Corporation's 8080 microprocessor was used for the implementation of the Kalman filter because it was available at the Naval Postgraduate School along with supporting software.

The interface between the microprocessor and the radar was not constructed due to: (1) Cost, time and manpower constraints, and (2) The performance of the system could not be effectively measured against non-cooperative targets of opportunity.

The performance of the radar and the interface equipment were effectively simulated on the Naval Postgraduate School's IBM-360 computer, thus permitting comparison of the microprocessor's performance against that of the IBM-360 system for known target tracks.

The output of the tracking system would be displayed for human utilization rather than being electrically interfaced to fire control equipment. The minimum standards of performance were established as that obtainable from a well trained, fully manned underway watch section typical of that employed on a modern destroyer not equipped with a Naval Tactical Data System (NTDS).

II. BACKGROUND

A. PERFORMANCE OF A TYPICAL CIC WATCH TEAM

Onboard naval ships not equipped with Naval Tactical Data Systems (NTDS), the tracking of radar contacts, both surface and airborne, is performed manually by the Combat Information Center (CIC) watch team. During normal peacetime steaming, this watch team may consist of from two to ten personnel, depending on the size of the ship and the complexity of its CIC equipment. Due to the risk of collision with other ships, their effort is primarily devoted to detecting, tracking and computing the course, speed, and closest point of approach (CPA) of other ships within a 20 to 30 nautical mile radius. This can be a tedious, error-prone task which generally requires a duplicate effort by the bridge watch team in order to avoid the awesome consequences of a collision at sea. The installation of an economical device to track and compute the tactical data for surface radar contacts could eliminate one or more watch-standers in CIC while improving the reliability of the information provided to the bridge watch team.

The manual method of computing the course, speed and CPA of a radar target involves a graphical solution of the true and relative velocity vectors of the radar target's motion and own ship's course and speed. The target's relative position measurements are obtained from a radar

repeater. The range measurement accuracy is at best ± 20 yards with a well calibrated repeater and a well trained operator. The bearing measurement accuracy is at best ± 1 degree with a well trained operator. The own ship's course and speed used in determining the own ship's velocity vector are normally based on the ordered course and speed. The course actually steered by the helmsman may vary from the ordered course by ± 2 degrees. Depending on the condition of the ship's hull and its engineering plant, the speed will be accurate within ± 1 knot. The combined results of these errors lead to a target course accurate to within ± 10 degrees and a target speed accurate to within ± 3 knots, not allowing for human error. The major disadvantages of the manual solution, in addition to its poor accuracy, is the time required to obtain the solution and the inability to obtain solutions if either the target or own ship maneuvers between radar position measurements, typically 3 or more minutes apart. The estimates of error are based on the author's personal observation of unalerted CIC watch sections.

B. THE TRACKING PROBLEM

The "Tracking Problem" as referred to in this investigation is that of using radar measurements to accurately determine the course and speed of a waterborne radar target. The targets of interest include merchantmen, warships, and other ships capable of open ocean operations. As such,

their motion on the surface of the ocean can be approximated by a linear dynamic system. In a sampled data system, such as a search radar where measurements are taken once each antenna rotation, the target dynamics can be represented in matrix form as [3]:

$$Z_{n+1} = \Phi Z_n + \Gamma a_n$$

$$Z_n = \begin{vmatrix} X_n \\ \dot{X}_n \\ Y_n \\ \dot{Y}_n \end{vmatrix}$$

$$\Phi = \begin{vmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$\Gamma = \begin{vmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{vmatrix}$$

Z_n = Vehicle State at Scan n

- X_n - E/W Postion
- $\dot{X}_n$ - E/W Velocity
- Y_n - N/S Position
- $\dot{Y}_n$ - N/S Velocity

T = Time Between Scan n and n + 1

a_n = Acceleration Acting on the Target from Scan n to n + 1 (E/W and N/S components)

If the radar measurements were perfectly accurate and the target had zero acceleration, then the tracking problem

would be a simple velocity vector problem (Figure 1). Unfortunately, the radar bearing and range to the target on any particular scan is contaminated by errors which are a function of the radar system and the received signal to noise ratio. The measurement errors, to be discussed in the next section, are characteristically Gaussian. An optimal estimator is well suited to tracking problems such as this.

"An optimal estimator is a computational algorithm that processes measurements to deduce a minimum error estimate of the state of a system by utilizing: knowledge of system and measurement dynamics, assumed statistics of system noises and measurement errors, and initial condition information." [4]

A Kalman filter is a common optimal filtering technique for estimating the state of a linear system [4]. It is chosen for implementation because it is computationally the most demanding as compared to other techniques used in similar tracking systems and, as such, places the most demand upon the microprocessor. The details of the Kalman filter will be covered in Chapter IV following an analysis of the measurements and associated hardware which provide its inputs and determine its parameters.

C. MEASUREMENTS

In order to solve the tracking problem, it is necessary to make measurements of (1) target range, (2) target bearing, (3) own ship's course, (4) own ship's speed, and (5) time

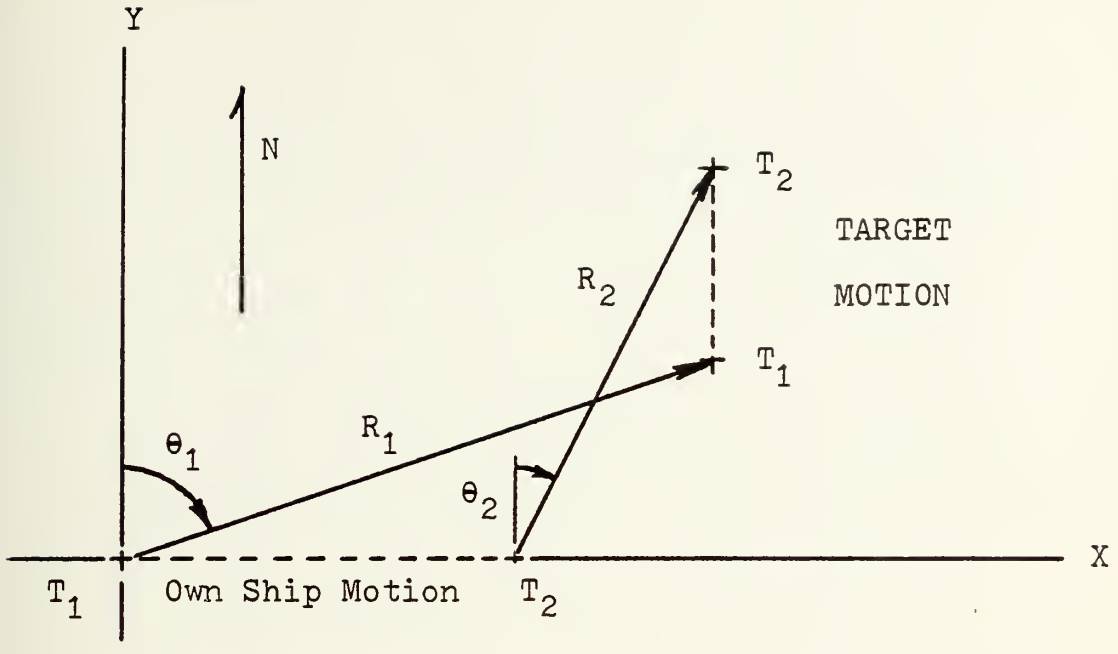


FIGURE 1a GEOGRAPHIC MOTION PLOT

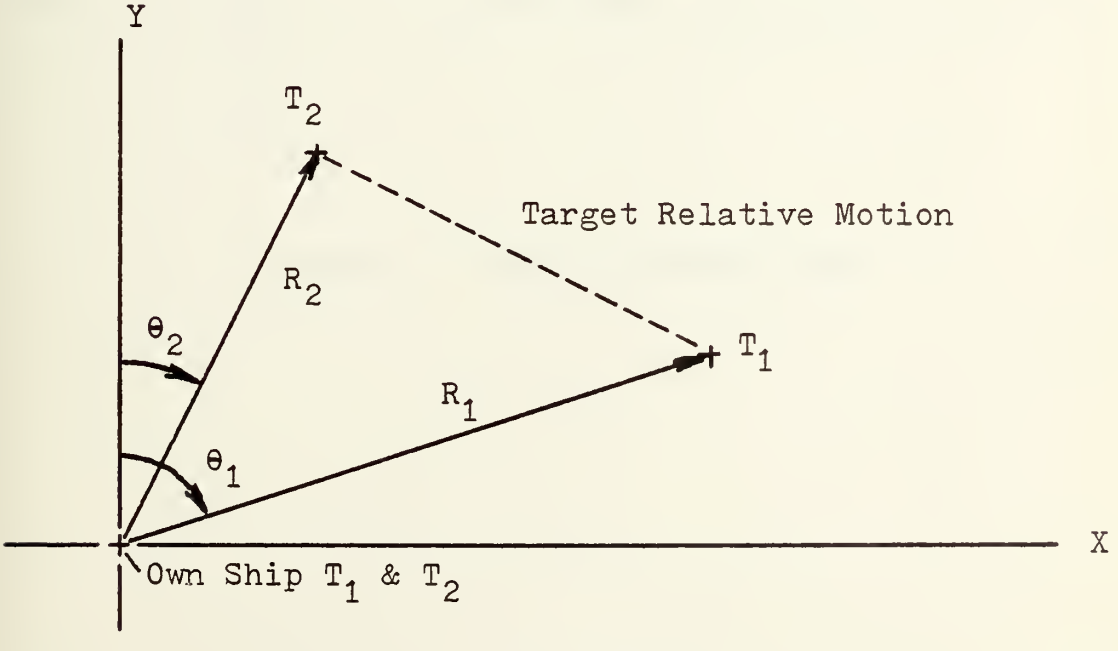


FIGURE 1b RELATIVE MOTION PLOT

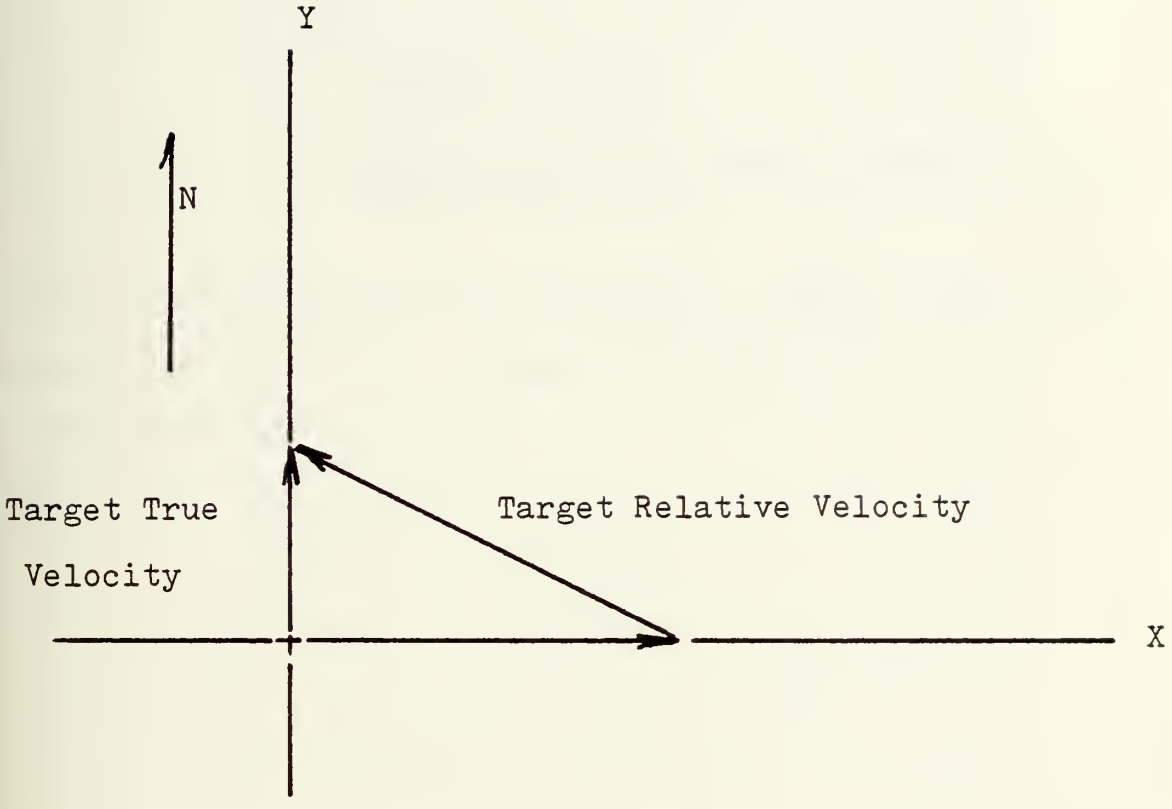


FIGURE 1c VELOCITY VECTOR PLOT

of the measurement. It is assumed that all noise is Gaussian and received signal to noise ratios are large ($>> 1$).

The target range is determined by measuring the time required for a radar pulse to travel to and from the target.

$$R = \frac{c\ t}{2}$$

R - Range to Target

c - Speed of Light

t - Elapsed Time from Pulse Transmission to Pulse Detection

Using threshold detection to obtain a leading edge measurement, the theoretical standard deviation of error for the measurement of t is [5].

$$\left[\overline{(\Delta t)^2}\right]^{\frac{1}{2}} = \frac{\tau_r}{(2\ s/n)^{\frac{1}{2}}}$$

Δt = Error in Elapsed Time Measurement t

τ_r = Rise Time of Received Pulse

s/n = Signal to Noise Ratio of Video Pulse

If the rise time of the pulse (τ_r) is limited by the receiver IF bandwidth B, then $\tau_r \approx \frac{1}{B}$ [5]. The theoretical standard deviation of range error using leading edge detection is therefore:

$$\sigma_R = \frac{c}{2B(2\ s/n)^{\frac{1}{2}}}$$

σ_R = Standard Deviation of Range Error

The mean range error is negligible when compared to the range measurement. Additionally, the mean error will tend

to be cancelled by subtraction in the relative velocity calculations. Thus the computed target course and speed will not be affected by the mean range error.

The target bearing is determined by measuring the bearing of the radar beam at the time the target video is detected. The mean error of the bearing measurement will be a function of the antenna beam pattern, and the target range and radar cross section. The resulting bearing to target will not be through the center of the target but the computed target course and speed will not be affected.

Using beam splitting, a technique for estimating the center bearing of the target, the theoretical minimum standard deviation of bearing error for a Gaussian beam distribution is [5]:

$$\sigma_\theta \approx \frac{1.06\ \theta_g}{\left[N_g (s/n)_c\right]^{1/2}}$$

where

σ_θ = Standard Deviation of Bearing Error

θ_g = Two Way Beamwidth (= 0.4250 θ_B, θ_B = Half Power Beamwidth) (Same units as σ_θ)

$(s/n)_c$ = Signal to Noise Ratio at Center of Beam

N_g = Number of pulses transmitted as the antenna rotates through 2 θ_g

The standard deviation of both range error and bearing error are functions of the video signal to noise ratio. The video signal to noise ratio for a radar with single pulse detection is given by [5]:

$$\frac{S}{n} = \frac{P_{pk}\, G^2\, \lambda^2\, \sigma\, 10^{-\alpha R/5}}{(4\pi)^3\, k\, T_o\, F_n\, B_n\, L_s\, R^4}$$

where

P_{pk} = Peak Transmitted Power (watts)

G = Antenna Gain

λ = Wavelength (meters)

σ = Radar Cross Section of Target (m^2)

α = Attenuation Constant of Propagation Medium (db/m)

R = Range to Target (meters)

k = Boltzmann's Constant (1.23×10^{-23} joule/deg)

T_o = Standard Temperature (290° K)

B_n = Receiver Noise Bandwidth (HZ)

L_s = System Losses

When these relationships are applied to the AN/SPS-10 (series) Surface Search Radar, with the AS-1161/SPS Antenna, the standard deviations for bearing and range can be related to target range and cross sectional area as shown in Figures 2 and 3. The minimum performance characters of this radar system are listed below.

Peak Power = 190 kw [6]

Antenna Gain = 30 db [8]

Pulse Repetition Rate = 625 HZ [6]

Frequency = 5825 MHZ [6]

Propagation Attenuation = $.6 \times 10^{-5}$ db/m [5]

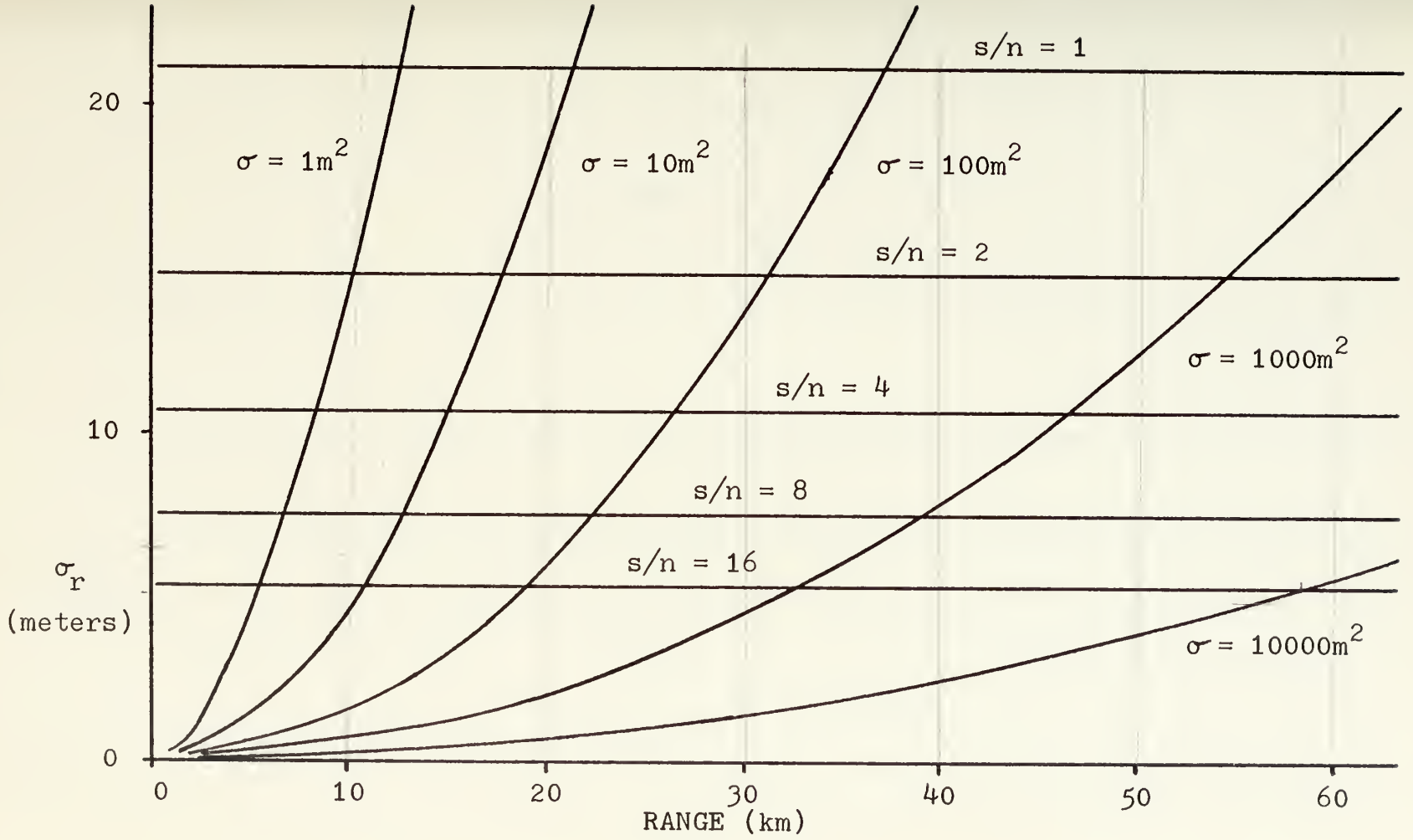


FIGURE 2 STANDARD DEVIATION OF RANGE ERROR VS. RANGE

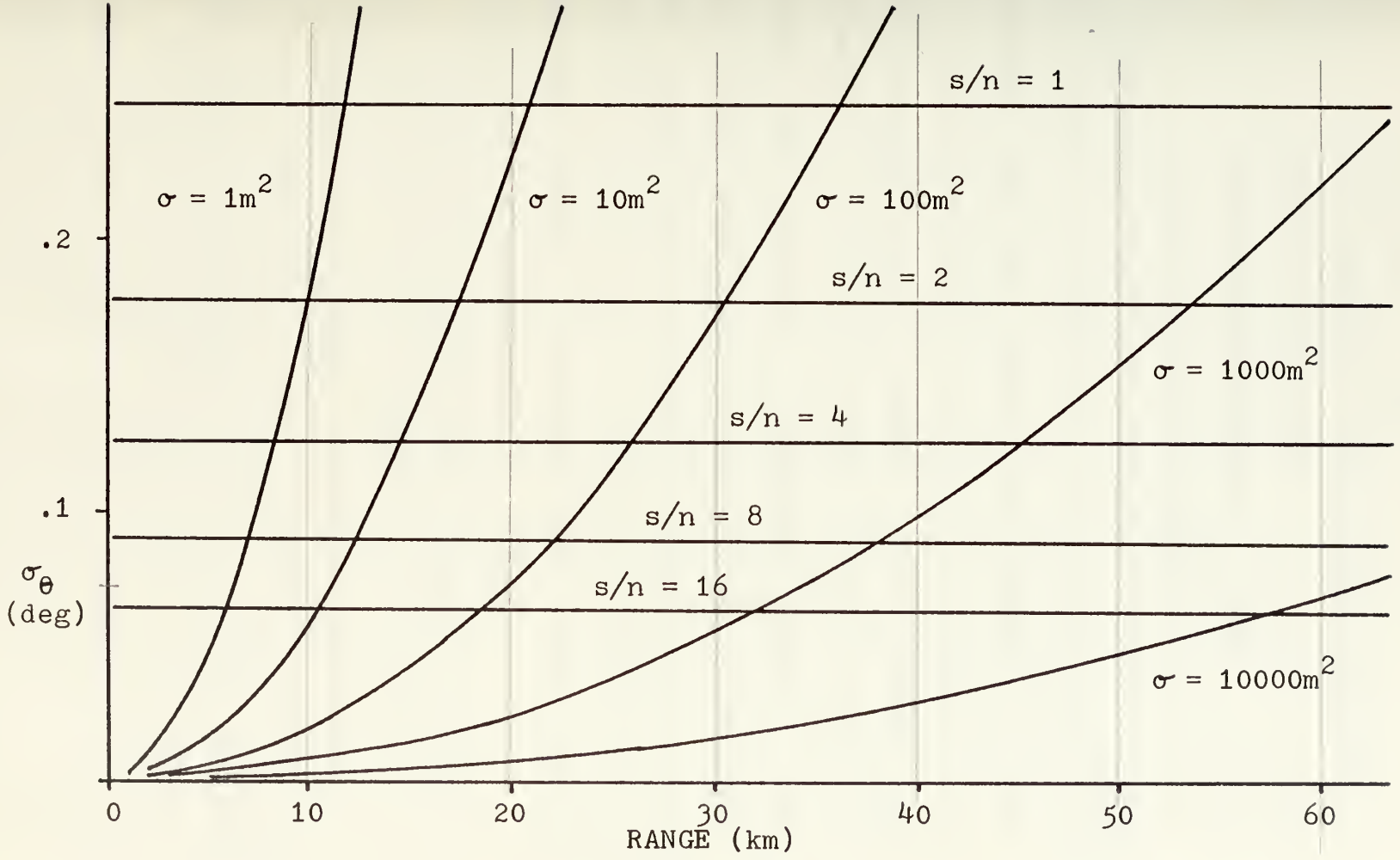


FIGURE 3 STANDARD DEVIATION OF BEARING ERROR VS. RANGE

Noise Figure = 22.9 db [7]

Bandwidth = 5 MHZ [7]

Noise Bandwidth (1.05 x B) ≈ 5.25 MHZ [5]

System Losses ≈ 5 db [5]

Antenna Rotation Rate = 17 RPM [8]

Antenna Horizontal Beamwidth = 1.9° [8]

As proposed earlier, a threshold detector could be used to indicate the occurrence of the leading edge of the received radar video pulse. The threshold to noise ratio of such a detector will establish the ranges at which various targets can be tracked and the minimum signal to noise ratio which the tracking system will encounter. Leading edge detection is preferred because it eliminates errors resulting from variations in the pulse width. When considering the threshold to noise ratio of such a detection scheme, it is necessary to establish the desired mean time between false alarms, and the desired minimum probability of detection of the target.

The mean time between false alarms is a measure of how often the tracking system will erroneously recognize a noise pulse as a target. The tracking system can not function effectively if this occurs frequently. A mean time between false alarms of 60 sec has been selected.

The mean time between false alarms is a function of the signal to noise ratio and the IF bandwidth of the receiver [5].

$$T_{FA} = \frac{1}{B_{IF}} \text{EXP}\left(\frac{V_T^2}{2\Psi_o}\right)$$

$\left(\frac{V_T^2}{2\Psi_o}\right)$ = Threshold to Noise Power Ratio

B_{IF} = IF Bandwidth

T_{FA} = Mean Time Between False Alarms

The exponential term in this expression is the inverse of the probability of false alarm (P_{FA}) [5].

$$P_{FA} = \text{EXP}\left(-\frac{V_T^2}{2\Psi_o}\right)$$

Thus

$$P_{FA} = \frac{1}{T_{FA}\, B_{IF}}$$

The mean time between false alarms as seen by the tracking system can be increased if the tracking system is disabled whenever targets are not anticipated. For the AN/SPS-10, the maximum unambiguous range is 129.6 nm of which only 30 nm is of interest to the tracking system. Therefore, the measurement circuitry could be disabled 77 percent of the time. This would multiply the mean time between false alarms by 4.32. An additional advantage can be gained by using bearing and range gates to further limit the opportunity for false alarms. If such bearing and range gates isolated 1000 yard squares for each target and the tracking system were tracking eight targets, the mean time between false alarms would be increased by an additional factor of 1414. The combination of these factors leads to a probability of false alarm of 2.036×10^{-5} and

a threshold to noise power ratio of 10.8 or 10.3 db.

The probability of detection of the target is a measure of how often the tracking system will recognize the target video when a target is present. It is essential to the effectiveness of the tracking system that it detect the target video reliably. The probability of detection is a complex function of the signal to noise ratio, the characteristics of the detector, and the probability of false alarm requiring numerical techniques [5]. Figure 2.7 of Reference 5 shows a plot of the probability of detection of a sine wave in noise as a function of the signal to noise power ratio for various probabilities of false alarm. This plot shows that a signal to noise ratio of 12.5 db to 14.0 db is required for a probability of detection of 0.90 to 0.99, respectively, and a probability of false alarm of 2.0×10^{-5}. If all targets were perfect radar reflectors, it would be possible to predict the performance of the threshold detector and the range at which various targets could be tracked. However, the reflected radar signals from the targets of interest will fluctuate due to cancellation and reinforcement of the waves reflected from various portions of the target. These fluctuations can be modeled as either Swerling's Case I or Swerling's Case II [5]. In either case, an additional signal to noise ratio of 8 db to 17 db will be required to insure a probability of detection of 0.9 to 0.99, respectively [5].

Thus the proposed threshold detection scheme should have a threshold to noise power ratio of 10.8 to achieve an effective mean time between false alarms of 60 sec. This would lead to a worse case minimum average signal to noise ratio of 20.5 db to 31.0 db for a probability of detection of 0.9 to 0.99, respectively.

For a radar measurement system with the parameters established above, targets with the following average radar cross sections could be tracked with the AN/SPS-10 at the ranges indicated.

Average Radar Cross Section (m^2)	Range (yards/meters)
1	3,600/3,290
10	6,400/5.850
10^2	11,500/10,520
10^3	20,000/18,290
10^4	35,000/32,000
10^5	60,000/54,860

Based on a threshold to noise ratio of 10.8, the standard deviation of the radar bearing and range error (Figures 2 and 3) will be 0.0763 degrees and 7.057 yards, respectively.

The own ship's course and speed measurements come from the ship's gyro and pitometer log, respectively. Accordingly, their error statistics are a function of the equipment installed on the particular ship. This equipment is frequently associated with the ship's fire control equipment

and, as such, is intended for tracking operations.

The gyro has a dual influence on the tracking problem. It not only determines own ship course but also determines the true bearing of the radar antenna. The mean error in radar bearing, and likewise own ship's course, will result in an error in the solution. This can be viewed as a shift in the reference plane which does not prevent the correct prediction of the future relative position of targets. As a refinement to the tracking system, it would be better to measure the average course and speed made good since the last measurement rather than the instantaneous values at the time of the measurement. Such an average measurement could be achieved by continuously integrating the variations in the resolved N/S and E/W velocities using a time constant equal to one-half the period of the antenna rotation rate. This integration process would have the desirable side-effect of minimizing the variance of error in own ship's course and speed. For subsequent calculations, it is assumed that the standard deviation of error of own ship's gyro is 0.1 degrees and own ship's speed indicator is 0.1 knots.

The fifth and final measurement, the time of the measurement, could be made with a very high accuracy, depending on the quality of the clock used. The objective is to determine the elapsed time between position measurements. The antenna rotation rate determines the mean elapsed time between measurements on a unique target, and the pulse

repetition rate of the radar determines the useful precision to which the elapsed time needs to be measured. The mean rotation rate for the AS-1161/SPS Antenna is 17 RPM. The mean pulse repetition rate for the AN/SPS-10 (series) Radar is 637.5 HZ. Thus, the mean elapsed time between measurements is 3.53 sec and the useful precision of the time measurement is $\pm$ 0.784 m sec.

D. MICROPROCESSORS

A microprocessor is a large scale integrated (LSI) semiconductor device or family of devices which contain an arithmetic logic unit (ALU) and several working registers capable of performing binary or BCD arithmetic and/or Boolean logic operations. Microprocessors generally fall into one of five categories; (1) single-chip controllers, (2) 4-bit family, (3) 8-bit family, (4) 16-bit family, and (5) bit slices [9].

The single chip controllers are LSI devices which incorporate all the necessary hardware, including program and data memory circuitry, on a single chip. Their data processing applications include devices such as calculators, gas pumps, cash registers, and scales. They include devices with 4-bit and 8-bit ALUs. Typically the 4-bit versions perform serial BCD operations. They are limited in their speed of program execution by the power of their instruction sets and the size of their ALUs. The complexity of the operations they can perform is limited by the size of

their memories. Some manufacturers are developing single-chip versions of their 8-bit family devices which will expand the capabilities of the single-chip controllers to the point of overlapping the 8-bit families [9].

The 4-bit microprocessor families are similar in function to the single-chip controllers except the memory is not included on the chip. Therefore, the 4-bit families are capable of handling larger programs. The term family refers to the set of compatible chips which go together to make up the microprocessor. They require greater design effort than the single chip controllers because several chips, including clocks and I/O interface chips, along with the separate memory, are required for even the simplest task. Like the 4-bit controllers, the 4-bit microprocessor families are limited in the speed of their program execution by the power of their instruction sets and the size of their ALUs.

The 8-bit families are presently dominating the microprocessor applications [9]. Their instruction sets are approaching the complexity of those used in minicomputers. They typically provide for 8-bit data manipulation along with 16-bit memory addressing. "---INTEL Corporation's 8080 family, with its enhanced 8080A CPU, Motorola Semiconductor's 6800 family, with its enhanced 6800D CPU, and Rockwell's PPS-8 family currently rank one, two, and three in popularity among users." [9]

The strength of the 8080 lies in the power of its

instruction set and the extent of its software support. The 8080 has 72 basic instructions, including conditional branching, decimal as well as binary arithmetic, logical, register to register, stack control and memory reference instructions [1]. These instructions can be executed in 4 to 18 cycles of its 2 MHZ clock allowing some instructions to be executed in 2 μsec [1]. These instructions facilitate 16-bit addition and register to register transfers. Among its software support are a System Monitor, an Assembler, a Text Editor, and a PL/M compiler, cross assembler, and simulator. The popularity of PL/M, an INTEL derivative of PL/1, and the 8080 have caused other manufacturers to model their microprocessors after the 8080, such as Advanced Micro-Devices 9080A, or to develop their own version of PL/M such as PL/M 6800 by Intermetrics for the Motorola's 6800.

The 8-bit devices are limited in performance by their 8-bit data path to and from memory. This necessitates using several memory cycles to read or write address and other 16-bit variables.

The 16-bit microprocessor family, including National's PACE, Texas Instruments Inc.'s 9900, and General Instrument Corp.'s CP1600, attempt to reduce the number of instructions and machine cycles required to manipulate memory addresses (typically 16-bit variables), and data originating from A-D converters or going to D-A converters (typically 10-16 bits). Their performance approaches that of minicomputers.

Western Corp.'s MCP 1600 was originally designed for DEC's LSI-11 16-bit minicomputer and was subsequently distributed as an independent device [9].

The bit-slice architectures, such as Advanced Micro Devices Inc.'s AM 2900, INTEL Corp.'s 3002, and Texas Instrument Inc.'s SBP0400, are taking on the tough process-control and high speed controller jobs previously handled by minicomputers. A typical bit-slice device is a 2 or 4 bit vertical slice of a CPU which is horizontally expandable in 2 or 4 bit increments to form a 16 bit, 32 bit or other multiple of 2 or 4 bit CPU. They utilize Schottky TTL, I^2L or ECL circuitry and thus are much faster than the previously mentioned microprocessors which are primarily MOS devices. The bit-slice architecture offers the ultimate in design flexibility while demanding the most effort by the designer. The design effort may include an extensive amount of microprogramming prior to development of the working program. Additionally, bit-slice devices are the most expensive form of the microprocessor.

In summary, the 8-bit and 16-bit microprocessors are the best alternatives to the tracking problem. The single-chip controllers and 4-bit families have insufficient memory and/or are too slow to handle the Kalman filter algorithm. The bit slice architectures have excess capability while requiring unnecessary design and hardware cost. As will be seen later, the tracking problem can be solved accurately with 16-bit resolution in the arithmetic operations. Thus

an 8-bit microprocessor programmed for double precision arithmetic or a 16-bit microprocessor program for single precision arithmetic is suitable.

III. SYSTEM INTERFACES

The required interface equipment can be grouped into three categories; (1) the radar interface, (2) the own ship course/speed interface, and (3) the operator/user interfaces. Although the interface hardware was not constructed, it is essential to the validity of the proposed tracking system to verify the feasibility of making the required radar and own ship measurements within the proposed tolerances.

A. RADAR INTERFACE

The radar interface serves four functions; (1) to convert the radar video to rectangular pulses compatible with the logic circuitry in the range measurement subsystem, (2) to identify the radar video of the target whose position is to be measured, (3) to measure and store the bearing, range and time of measurement of the designated radar target, and (4) to transmit a signal to the own ship course/speed and the operator/user interfaces to indicate when the radar measurements have occurred.

The conversion of the radar video to a rectangular pulse compatible with the bearing and range measurement circuitry could best be accomplished at the threshold detector. The threshold detector could consist of one of a variety of LSI voltage comparators currently available. The input video would be in the 0-2 volt range depending on the setting of

the radar IF amplifier gain adjustment. The threshold voltage would be a technician adjustment made by first setting the IF amplifier gain to obtain a good video display on the radar's master repeater and then adjusting the threshold voltage on the comparator to just detect a known signal. The critical parameters of the threshold detector are a bandwidth greater than that of the IF amplifier (5 MHZ) and a high noise immunity.

Target identification could be accomplished either with software or hardware. A software solution would result in the measurement and processing of a large number of targets which the operator had no interest in and would place an unnecessary burden on the microprocessor. Thus a hardware technique is preferred. A simple solution to this problem is the use of bearing and range gates to enable the measurement circuitry at the time a particular target video is anticipated.

The width and time of occurrence of the bearing and range gates can be determined in several ways, all of which are a function of the accuracy with which the future location of the particular target can be estimated. This estimate is a function of (1) the maneuverability of the target, (2) the elapsed time between measurements and (3) the accuracy of the computed course and speed of the target.

The majority of targets of interest will be capable of turning radii of 100-1000 yards, velocities of 0-40 knots, and accelerations of 0-.5 yards/sec^2. The antenna rotation

rate determines the mean time between measurements, which in the case of the AS-1161/SPS antenna equates to 3.53 seconds [8]. The accuracy of the computed course and speed of the target will depend on the tracking system parameters and the maneuverability of the target.

After the three major factors have been accounted for, the probable measurement error should be added to account for differences in the target's predicted bearing and range from the measurement system bearing and range values. Because of the uncertainty of these many factors, it is necessary to use wide bearing and range gates to insure that the target is not lost. The unfortunate outcome of using wide bearing and range gates is the increased likelihood that two adjacent targets will enter each others bearing and range gates and become confused.

The best solution to this dilemma is to allow the operator to select a target classification from his control station which would determine which of several predetermined bearing and range gate widths or width determining algorithms would be used with the designated target. As will be shown later, the designation of target maneuverability by the operator is not only useful in the measurement circuitry but also enhances the Kalman filter.

The determining factor in the accuracy to which the bearing, range and time measurements can be represented in the microprocessor is the value represented by the least significant bit in the binary data word. If the

digitizing process is carried out with the digital zero equal to the analog zero it will introduce a mean error equal to one-half of the value represented by the least-significant bit and a standard deviation of error equal to $3^{-\frac{1}{2}}$ times the value represented by the least significant bit. If the digital zero is offset by minus the previous mean error, then the mean error will be zero and the standard deviation of error equal to $12^{-\frac{1}{2}}$ times the value represented by the least significant bit. Obviously, the zero offset approach is preferred.

The length of the binary data word used to represent the bearing, range, and time measurements is determined by the value represented by the least significant bit, and the maximum value which is to be represented.

For the system under consideration, the least significant bit should represent the standard deviation of error of the target bearing with gyro error (0.1258 degrees or 0.002196 radians) and the target range (7.0574 yards), and the useful precision of the time measurement ($\pm$ 0.784 msec), as a minimum. The maximum value of each measurement should be 2π radians, 60,000 yards and approximately 8.0 secs. This would require 11.5 bits for the bearing measurement, 13.1 bits for the range measurement and 13.3 bits for the time measurement. If an 8-bit microprocessor is to be used, then data words of 8, 16, or 24 bit lengths are practical. In addition, the INTEL 8080 microprocessor has some instructions for manipulating 16-bit data words.

Thus 16-bit data words were considered optimum. This results in the following data format:

Measurement	Range	MSB	LSB
Bearing	0-2π rad	π rad	0.0959 mrad
Range	0-60,000 yds	30,000 yds	0.9155 yds
Time	0-8.0 sec	4.0 sec	0.1221 msec

By changing the most significant bit to the next even power of two, the units can be rationalized as follows:

Measurement	Range	MSB	LSB
Bearing	0-2π rad	4.0 rad	2^{-13} rad
Range	0-65,535 yds	32,768 yds	1.0 yds
Time	0-7.999 sec	4.0 sec	2^{-13} sec

The resolution resulting from this format will result in less than 1.0 percent increase in the standard deviation of error as a result of digitizing the data. The remaining question is what resolution is possible with the available hardware.

The bearing measurement could be conveniently made by combining one of the synchro-to-digital converter modules such as Transmagnetics series 1651, which has 14-bit resolution, and two 16-bit latchs, enabled by the video circuitry, to store the first and last bearings from which video was detected from a particular target on each scan. The microprocessor software could then compute the center bearing to the target. An alternate scheme

would be to use a single latch to store the first bearing from which video was detected from a particular target on each scan and a counter to count the number of video pulses from the target on each scan. The microprocessor software could then use this data to estimate the center bearing. A third and far less accurate scheme would be to simply use the bearing of the first video pulse. The first scheme is recommended.

The range measurement is essentially a time interval measurement. A range of 1 yard corresponds to 6.1 nsec. Thus a binary counter would have to be clocked at 163.9 MHZ to count radar range in intervals of 1 yard. Ordinary Transistor-Transistor-Logic (TTL) counters are limited to clock frequencies of 30 MHZ and below, while high-speed TTL counters can operate as high as 50 MHZ [10]. Schottky-Clamped TTL counters can operate reliably at clock frequencies of 100 MHZ [10]. Emitter-Coupled-Logic (ECL) circuitry could perform at higher clock frequencies but the complexity and cost of the circuitry would increase considerably due to the unavailability of ECL Medium-Scale Integrated (MSI) semiconductor chips with the desired functions. A Shottky-Clamped TTL Counter, such as the 74S197, is suggested, with a compromise reduction in the clock frequency to 81.96 MHZ resulting in a minimum range resolution of 2 yards. An alternate scheme would be to mix ECL circuitry with the Shottky counters.

Another consideration in the range measurement counter

is the propagation of carry pulses through the counter. The 74S197 binary counter has a maximum propagation time of 13 nsec through each stage. For the 15 stages required for a maximum range of 65,536 yards, a total propagation time of 195 nsec would be required for a carry propagation to the highest significant bit. This is 16 times the clock period, which would lead to unstable data at the counter outputs. The easiest solution to this dilemma is to stop the clock input to the counter at the time the target video is detected and store the counter value after it has stabilized. This would necessitate a separate range counter in the interface hardware for each target being tracked.

The time measurement interface is comparatively straight forward. A 16-bit counter driven by a 8,192 HZ clock, with a storage latch for each target will suffice. As in the range counter, the carry propagation delay must be taken into account when storing the time measurement. Due to the slower clock frequency, the counter output will be stable 99 percent of the time. Simple combinational logic circuitry could be employed to synchronize the time measurement with the clock to avoid storing unstable counter outputs.

Using currently available hardware, the radar interface can be accomplished with resolutions well within the radar measurement errors specified earlier in this section. The following data formats are recommended.

Measurement	Range	No. of Bits	Resolution
Bearing	0-2π rad	14	.3835 mrad
Range	0-65,535 yards	15	2.0 yards
Time	0-8 sec	16	.1221 msec

B. OWN SHIP COURSE AND SPEED INTERFACE

The own ship course and speed interface serves two functions; (1) to smooth the variations in own ship's course and speed, and (2) to measure and store the smoothed course and speed to be associated with a particular target. There are two options of how to perform these functions, stemming from the fact that the microprocessor software uses own ship's course and speed data solely to compute own ship north/south and east/west velocities.

The first approach would be to process the course and speed measurements independently. The course measurement could be smoothed either mechanically or with an over-damped servo feedback system. A synchro to digital angle converter could be employed to digitize the measurement. The speed measurement could be smoothed electrically and digitized using an analog to digital converter.

The second approach would be to combine the course and speed measurements into north/south and east/west velocities. The course synchro signal could be converted to two analog signals corresponding to the sine and cosine of own ship's course using a synchro to DC sine/cosine converter, such as Transmagnetics model 655N V (series). These analog

signals could then be multiplied by an analog speed signal with two DC analog multipliers. Low-pass filters could be used to smooth the two signals and then digital to analog converters used to digitize them. This approach would simplify the work of the microprocessor but additional errors would be generated in the analog multipliers.

A compromise approach would be to use the first system but substitue a synchro to digital sine/cosine converter in place of the digital angle converter. With this technique, the microprocessor would receive smoothed north/south and east/west course component measurements and smoothed speed measurements.

Assuming a standard deviation of error in the course measurement of 0.1 degree (1.745×10^{-3} radians) prior to digitizing, a digital data format of 11.8 bits would be required for minimum resolution. Using a data format similar to that used for the target bearing (MSB = 2.0 radians) and 14-bit resolution, the standard deviation of error will only increase 0.3 percent as a result of digitizing. Assuming a standard deviation of error in the speed measurement of 0.1 knot prior to digitizing, and a maximum speed of 100 knots, a digital data format of 10 bits would be required. Using a 12-bit analog to digital converter or a 12-bit synchro to digital converter with the most significant bit representing 64 knots, the standard deviation of error will increase only 1.6 percent, as a result of digitizing.

In summary, the own ship course and speed interface could be accomplished using available hardware, with the following data formats and measurement precision.

Measurement	Range	No. of Bits	Resolution
Course	0-2π rad	14	.3835 mrad
Speed	0-128 knots	12	.03125 knots

C. OPERATOR/USER INTERFACE

The operator/user interface serves two functions; (1) to provide operator control of the tracking system, and (2) to display the output data to the users.

The operator should have control over (1) which radar video is to be tracked, (2) the number of targets being tracked, and (3) the tracking parameters to be used with each target.

It is proposed that the operator would control the system in the following manner. The measurement interface circuitry would be divided into tracking elements, each capable of handling one target. Each element would be composed of a bearing and range gate generation circuit, a range measurement circuit, and storage circuits for the bearing, range, time, own ship's course and own ship's speed data.

The operator's control panel would have an off/on/start switch for each tracking element. The system would be turned on with all tracking elements turned off. To track a target, the operator would place his radar repeater's

bearing and range cursor on the target video to be tracked and momentarily place the tracking element switch in the start position when the radar sweep passed the target and then place the tracking element switch in the on position. This process would momentarily couple the radar repeater's bearing and range cursor video to the tracking elements bearing and range gate circuit and enable the tracker's measurement circuitry, thus taking a measurement of the target's position. If the measurement circuitry detected target video, an indicator light would be lit. To stop tracking a target, the operator would simply turn the tracking element off. A control data word (8 bits) would be periodically read by the microprocessor to advise it as to which tracking elements have updated measurement data available. The operator would have a second multi-position switch which would allow him to select the bearing and range gate widths and the variance of acceleration parameters to be used with each target.

The output data to be displayed would include (1) target designation, (2) target bearing, (3) target range, (4) target course, and (5) target speed, as a minimum. For more sophisticated systems the target's closest point of approach (CPA) could be computed and displayed as (1) CPA bearing, (2) CPA range and (3) time of CPA. The speed of the data output is critical to the capability of the system to track multiple targets. Direct memory access techniques are suggested as the only practical solution.

After the display data is extracted from the tracking system, several display techniques could be used. Tabular displays in the form of 7-segment LED modules or cathode ray tubes would facilitate displaying the data to several users at the operators station and also transmission of the data to remote stations, such as the bridge, CO's cabin, and weapons stations. One of the major advantages of an automatic tracking system is its capability to instantly and continuously disseminate tactical data to remote stations thus eliminating the need for passing this information on sound powered phones as is currently the practice on non-NTDS ships.

IV. KALMAN FILTER

As stated earlier, the Kalman filter is a common optimal filtering technique for estimating the state of a linear system. "Among the presumed advantages of this type of data processor are that it minimizes the estimation error in a well defined statistical sense and that it utilizes all measurement data plus prior knowledge about the system. The corresponding potential disadvantages are its sensitivity to erroneous a priori models and statistics, and the inherent computational burden." [4]

A. DISCRETE KALMAN FILTER ALGORITHM

The rotation of the surface search radar antenna results in periodic measurements of the target positions. Accordingly, the discrete form of the Kalman filter, rather than the continuous form, is appropriate. The discrete Kalman filter can be described by a set of matrix equations.

Using the target state (Z_n) representation presented in the description of the tracking problem, in Section I B, the matrix form of the measurement process is [3]:

$$w_n = HZ_n + v_n$$

$$w_n = \begin{vmatrix} x_m(n) \\ y_m(n) \end{vmatrix}$$

$x_m(n)$ = Measured x Coordinate at Scan n

$y_m(n)$ = Measured y Coordinate at Scan n

$$H = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Z_n = Target State Vector at Scan n

$$v_n = \begin{vmatrix} v_x(n) \\ v_y(n) \end{vmatrix}$$

$v_x(n)$ = Random Measurement Error on $x_m(n)$

$v_y(n)$ = Random Measurement Error on $y_m(n)$

The estimated state vector at scan n ($\tilde{Z}_n$) is [3]:

$$\tilde{Z}_n = \Phi \hat{Z}_{n-1}$$

Φ - (See Tracking Problem Page 14)

$\hat{Z}_{n-1}$ = Optimal Estimate of Target State Vector at Scan n-1

The optimal estimate of the target state at scan n ($\hat{Z}_n$) is given by [3]:

$$\hat{Z}_n = \tilde{Z}_n + K_n(w_n - H\,\tilde{Z}_n)$$

$$K_n = \begin{vmatrix} A_{xx} & A_{xy} \\ B_{xx}/T & B_{xy}/T \\ A_{yx} & A_{yy} \\ B_{yx}/T & B_{yy}/T \end{vmatrix} = \text{Gain Matrix at Scan n}$$

The gain matrix (K_n) is computed from [3]:

$$K_n = \tilde{P}_n H^T (H \tilde{P}_n H^T + R_n)^{-1}$$

$\tilde{P}_n$ = Estimated Covariance of Estimation Errors Matrix Prior to Processing Scan n Measurements

$$R_n = \begin{vmatrix} \sigma_x^2(n) & \sigma_{xy}^2(n) \\ \sigma_{xy}^2(n) & \sigma_y^2(n) \end{vmatrix}$$ = Covariance of Measurement Error Matrix at Scan n

Following the processing of scan n measurements, an optimal estimate of the covariance of estimation error matrix ($\hat{P}_n$) for scan n can be computed [3]:

$$\hat{P}_n = (I - K_n H)\tilde{P}_n$$

I = Identity Matrix

From this, it is possible to compute the estimate of the covariance of estimation error matrix for scan n + 1 [3].

$$\tilde{P}_{n+1} = \Phi_{n+1} \hat{P}_n \Phi_{n+1}^T + \Gamma_{n+1} Q \Gamma_{n+1}^T$$

$Q = \sigma_a^2$ = Variance of Random Target Acceleration

The basis for the discrete Kalman filter in this form is the legitimacy of the target maneuver model. The variance of target acceleration must be equal and independent along the x and y axis, and from scan to scan. This is true for the targets of interest. Thus, this form of the discrete Kalman filter is valid if the position measurements were made in the x and y coordinates.

Using the polar coordinate system shown in Figure 4, the following transformations can be used with the radar's bearing (θ) and range measurements (R).

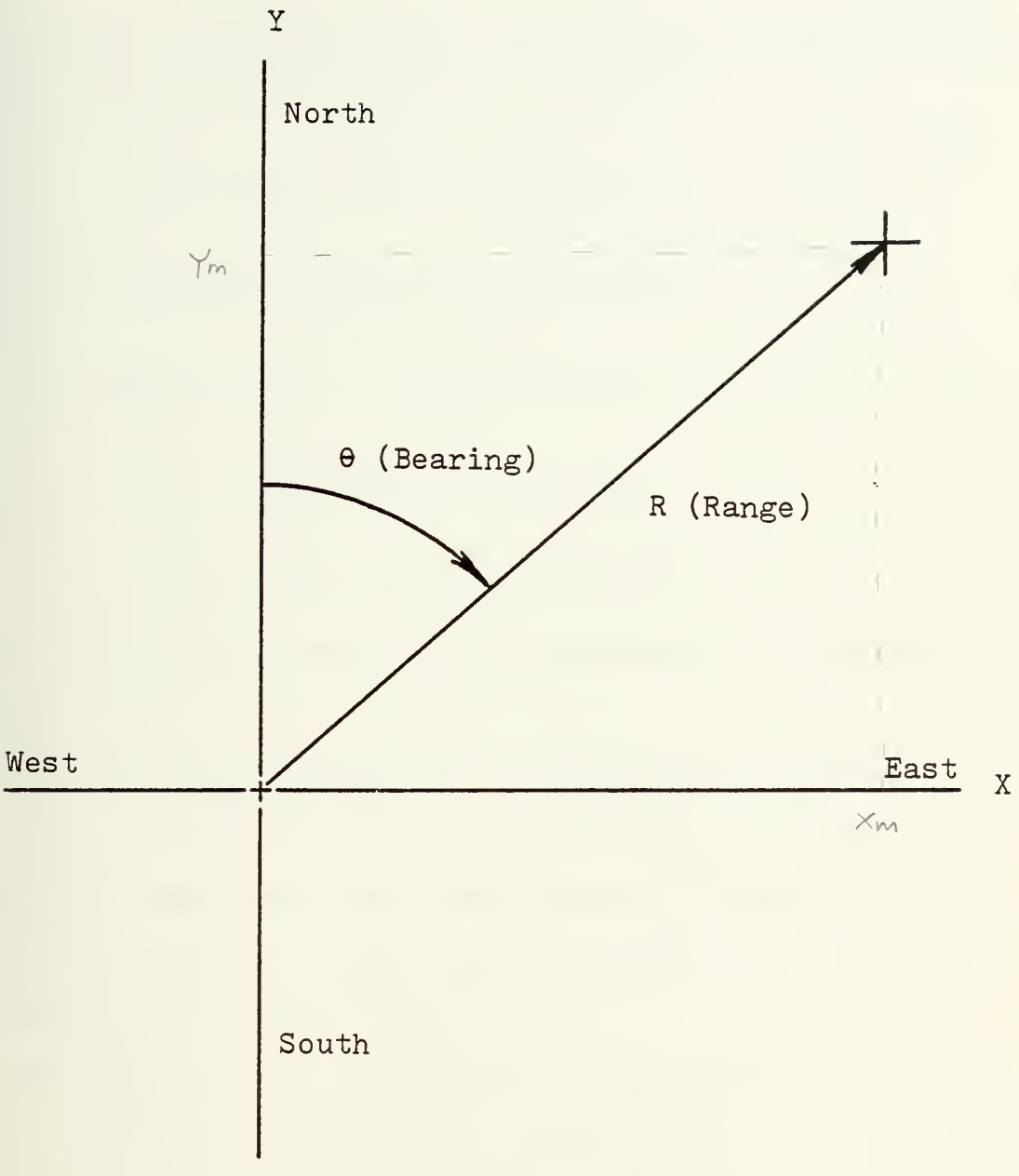


FIGURE 4 COORDINATE SYSTEM

$$x_m = R \sin \theta$$

$$y_m = R \cos \theta$$

Using these transformations, the elements of the covariance of measurement error matrix are [11]:

$$\sigma_x^2 = \sigma_r^2 \sin^2\theta + R^2 \sigma_\theta^2 \cos \theta$$

$$\sigma_y^2 = \sigma_r^2 \cos^2\theta + R^2 \sigma_\theta^2 \sin^2 \theta$$

$$\sigma_{xy}^2 = [\sigma_r^2 - r^2 \sigma_\theta^2] \tfrac{1}{2} \sin 2\theta$$

σ_r^2 = Variance of Range Error

σ_θ^2 = Variance of Bearing Error

This modification completes the description of the discrete Kalman filter.

In some applications, it is possible to precompute the values of the gain matrix and store them in the computers memory for subsequent use. This greatly simplifies the software algorithm. There are two reasons why this is not possible in this case.

The first is that the elapsed time between measurements (T) is not constant. This is primarily due to the fact that the radar antenna rotates relative to own ship's heading. Thus, changes in own ship's course can significantly increase or decrease the elapsed time between measurements (T).

The second reason is that the covariances of measurement

error (σ_x, σ_y, σ_{xy}) are a function of the bearing and range to the target. Consequently, the gain matrix must be computed on line for each target being tracked. A flow chart of these calculations is shown in Figure 5.

The solution of the Kalman filter equations in matrix form is unnecessarily complex for the microprocessor. Therefore, an algebraic solution for the value of each element in the various matrices was formulated. Appendix A lists this algebraic form of the Kalman filter equations.

B. KALMAN FILTER PARAMETERS

The performance of the Kalman filter is determined by three parameters; (1) variance of bearing error (σ_θ^2), (2) variance of range error (σ_r^2), and (3) variance of target acceleration (σ_a^2). The first two parameters are functions of the radar, the ship's gyro, and the interface system. The third parameter is a function of the target's relative motion and the desired smoothness of the output data.

The variance of bearing error is equal to the sum of the variance of the radar bearing error, (.0058 deg^2), the variance of the bearing-interface digitizing error (.00016 deg^2), and the variance of the ship's gyro error (0.01 deg^2). Thus, the total variance of bearing error is 0.016 deg^2 or .0000049 rad^2.

The variance of range error is equal to the sum of the variance of the radar range error (50 yd^2 or 42 m^2) and

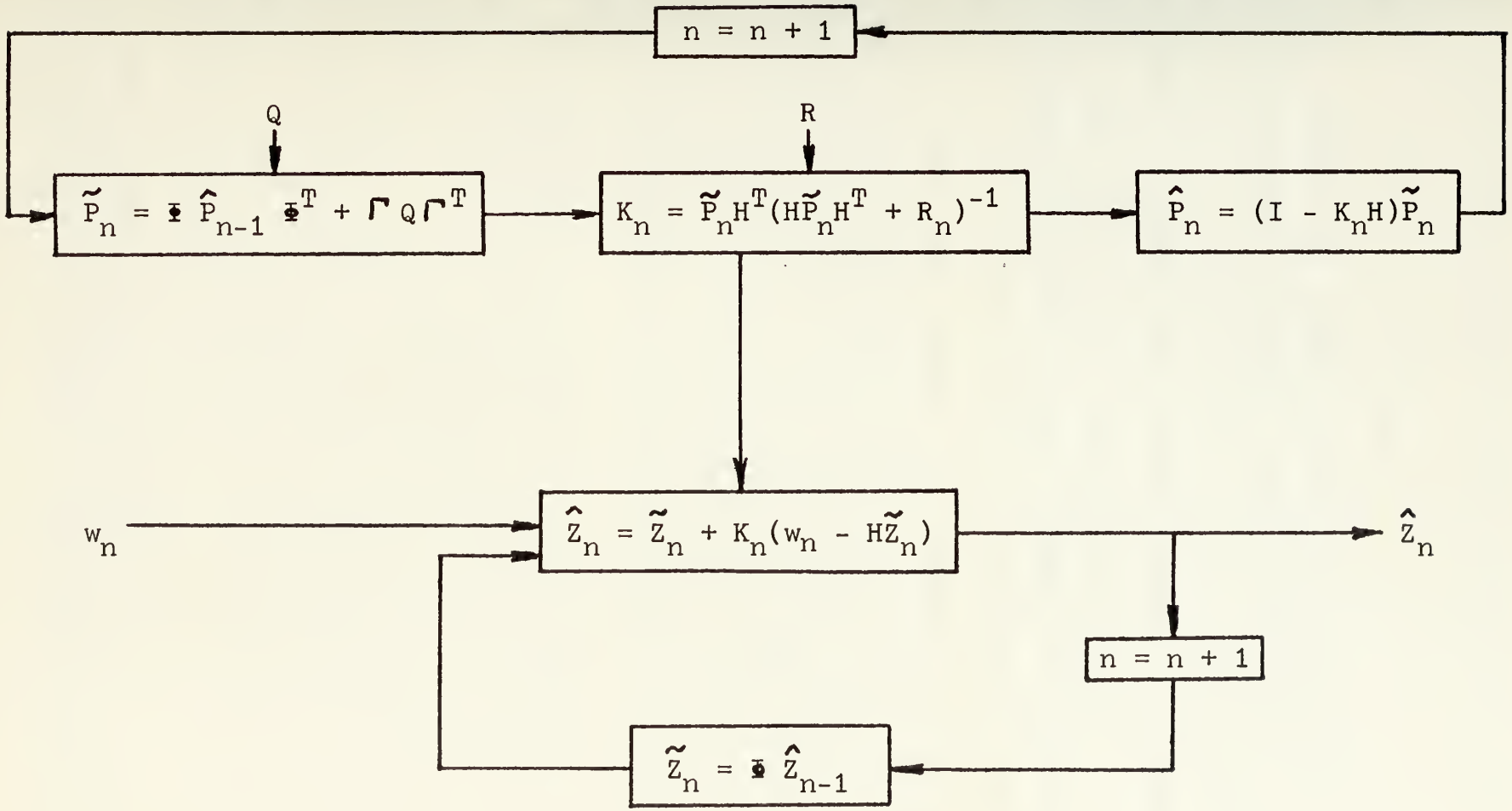


FIGURE 5 DISCRETE KALMAN FILTER FLOW DIAGRAM

the variance of the range interface digitizing error (1.3 yd^2 or 1.1 m^2). Thus the total variance of range error is equal to 51 yd^2 (43 m^2).

The variance of target acceleration parameter is a compromise between the user's desired smoothness of the output data for non-maneuvering targets and the ability of the filter to provide the closest estimate of a target's position and velocity for a maneuvering target.

A possible solution to this dilemma is to allow the operator to select the variance of acceleration parameter to be used with each target. This parameter, along with the bearing and range gate widths, could be selected by enabling one of several read-only-memories (ROMs) containing the desired parameters. The particular ROMs installed at any one time could be selected from a library of such ROMs by the Commanding Officer depending on the ship's mission.

The determination of what values of variance of acceleration to be used with various targets are best determined by modeling [4]. A computer model of the tracking problem was written and used to investigate the system performance with various values of variance of acceleration. The results of this investigation are presented in the figures discussed in Section IV E.

C. KALMAN FILTER INITIAL CONDITIONS

As illustrated by Figure 5, there are two basic loops in

the Kalman filter algorithm. They represent the recursive calculations of the optimal estimates of the covariance of estimation error matrix ($\hat{P}$) and the target state matrix ($\hat{Z}$). An initial condition must be established for each of these based on a priori information of the measurement process and the targets of interest.

The initial condition of the target state matrix is equal to the expected value of the initial state. The expected values of the initial position elements, X_0 and Y_0, are the first position measurements:

$$X_0 = R(0) \sin \theta(0)$$

$$Y_0 = R(0) \cos \theta(0)$$

The expected values of the initial velocity elements, $\dot{X}_0$ and $\dot{Y}_0$, are zero. Thus, the filtering of the target state does not begin until the second measurement.

The initial condition of the covariance of estimation error matrix is equal to the covariance of error of the initial state estimate.

$$\hat{P}_0 = E\left[\tilde{Z}_0 - \bar{Z}_0\right]\left[\tilde{Z}_0 - \bar{Z}_0\right]^T$$

$$\hat{P}_0 = \begin{vmatrix} E\left[(\tilde{X}_0 - \bar{X}_0)^2\right] & 0 & 0 & 0 \\ 0 & E\left[(\tilde{\dot{X}}_0)^2\right] & 0 & 0 \\ 0 & 0 & E\left[(\tilde{Y}_0 - \bar{Y}_0)^2\right] & 0 \\ 0 & 0 & 0 & E\left[(\tilde{\dot{Y}}_0)^2\right] \end{vmatrix}$$

The non-zero elements in this matrix are functions of the initial relative velocities which might be encountered,

and the measurement errors of the initial position. Assuming a uniform probability distribution of relative velocities from -50 yd/sec to +50 yd/sec ($\pm$ 46 m/sec), and an initial range of 50,000 yds (46,000 m), the variance of initial position error is 12,000 yds^2 (10,000 m^2) and the variance of initial velocity error is 830 yd^2/sec^2 (700 m^2/sec^2) for both the X and Y coordinates.

The primary influence of the initial conditions is the time required for the filter to achieve a steady-state solution. Large values lead to longer settling times for nonmaneuvering targets where small values lead to longer settling times for maneuvering targets.

D. SYSTEM COMPUTER MODEL

In order to evaluate the performance of the Kalman filter with various targets and parameters, a computer model of the system was derived. The computer model included the simulation of target and own ship motion, the radar measurement and interface process, the Kalman filter, and the computation of the filtered target course and speed. Additionally, the program included computation of measurement errors, target track errors, and associated statistics.

The computer model simulated the target and own ship motion by digitally integrating the north/south and east/west velocity components using an integration step of 0.01 sec and double precision arithmetic. Target and own ship course changes were accomplished by a second-order nonlinear

control system model with the ordered course as the input, a limit on the maximum angular rate, and the actual course steered as the output. The control system model parameters and angular rate limit were adjusted to simulate typical course changes for the targets of interest. Speed changes were accomplished by a first-order linear control system model with the ordered speed as the input and the actual speed as the output. No attempt was made to duplicate actual ship maneuvers, but only to approximate the maneuvers typical of the targets of interest.

The radar and interface systems were modeled by first comparing the rotation of the radar antenna to the bearing to each of the targets. Upon detection of a target, the bearing and range to the target were noted along with own ship's course and speed, and the time of detection. Random numbers representing the Gaussian distribution of measurement errors were added to the bearing and range measurement. The measurements were then truncated to simulate the digitizing process of the interface system.

The Kalman filter was implemented through FORTRAN statements similar to the algebraic expressions listed in Appendix A. The only modifications were those related to improving the efficiency of executing the computations.

The target's position, course, and speed were computed from the relative north/south and east/west positions and velocities provided by the Kalman filter, and the simulated measurement of own ship's course and speed.

The computer model was programmed in FORTRAN IV, compiled with a FORTRAN H Compiler, and executed on the Naval Postgraduate School's IBM 360/67 computer.

E. KALMAN FILTER PERFORMANCE

Using the computer model previously described, the performance of the Kalman filter was evaluated for two classes of targets, maneuvering and nonmaneuvering. In all cases, own ship was proceeding on a steady course (000^{o}) and at a fixed speed of 20 knots (37 km/h) from an initial position at the origin of the cartesian coordinate system. Eight targets were initially positioned and proceeding on the courses and at the speeds shown in Table 1. A unique set of random numbers was used with each target for the various computer runs. This eliminated the influence of variations in the random number distributions on the statistics of the Kalman filter output.

The following system parameters were used:

RADAR PARAMETERS	
Pulse Repetition Rate	100.0 pps
Range Precision	7.0574 yds (6.453 m)
Bearing Precision	0.1258 deg
Antenna Rotation Rate	17.0 rpm
DIGITAL INTERFACE PRECISION PARAMETERS	
Range LSB	4.0 yds (3.6576 m)
Bearing LSB	0.02197 deg

TARGET NO.	N/S POSITION (yards/meters)	E/W POSITION (yards/meters)	COURSE (degrees)	SPEED (knots/kms)
1	1500/1372	-1000/-914	135	12.5/23.2
2	-1500/-1372	1000/914	315	22.5/41.7
3	2830/2588	2830/2588	0	15.0/27.8
4	4240/3877	4240/3877	225	25.0/46.3
5	9900/9053	9900/9053	0	15.0/27.8
6	11310/10342	11310/10342	225	25.0/46.3
7	31110/28447	31110/28447	0	15.0/27.8
8	32530/29745	32530/29745	225	25.0/46.3

TABLE I INITIAL POSITION, COURSE AND SPEED FOR SIMULATED TARGETS

Time LSB	0.1221 msec
Own Ship Course LSB	0.02197 deg
Own Ship Speed LSB	0.03125 knot (0.05788 km/h)

The Kalman filter measurement error parameters (σ_θ^2 and σ_r^2) were selected to match the computed measurement errors resulting from the radar and interface parameters.

$$\sigma_\theta^2 = 4.870 \times 10^{-6} \text{ rad}^2 \ (0.01599 \text{ deg}^2)$$

$$\sigma_r^2 = 51.14 \text{ yd}^2 \ (42.76 \text{ m}^2)$$

The Kalman filter initial conditions were the same as those discussed in Section C of this chapter. The variance of target acceleration (σ_a^2) parameter was chosen as the independent variable.

Both scenarios began with the acquisition process for each of the eight targets, which involved setting the initial conditions of the Kalman filter, and ran for 5 minutes of simulated time. For the nonmaneuvering case, no changes were made throughout the run. For the maneuvering case, each of the targets executed a 45 deg course change beginning 120 sec after the start of the run. Targets 1, 4, 6 and 8 turned right, while targets 2, 3, 5 and 7 turned left. These turns were executed with an angular rate of 3.0 deg/sec. Figure 6 shows an example of the plot developed during such a left turn.

The actual measurement error statistics, based on 680 measurement samples from the computer model, for the

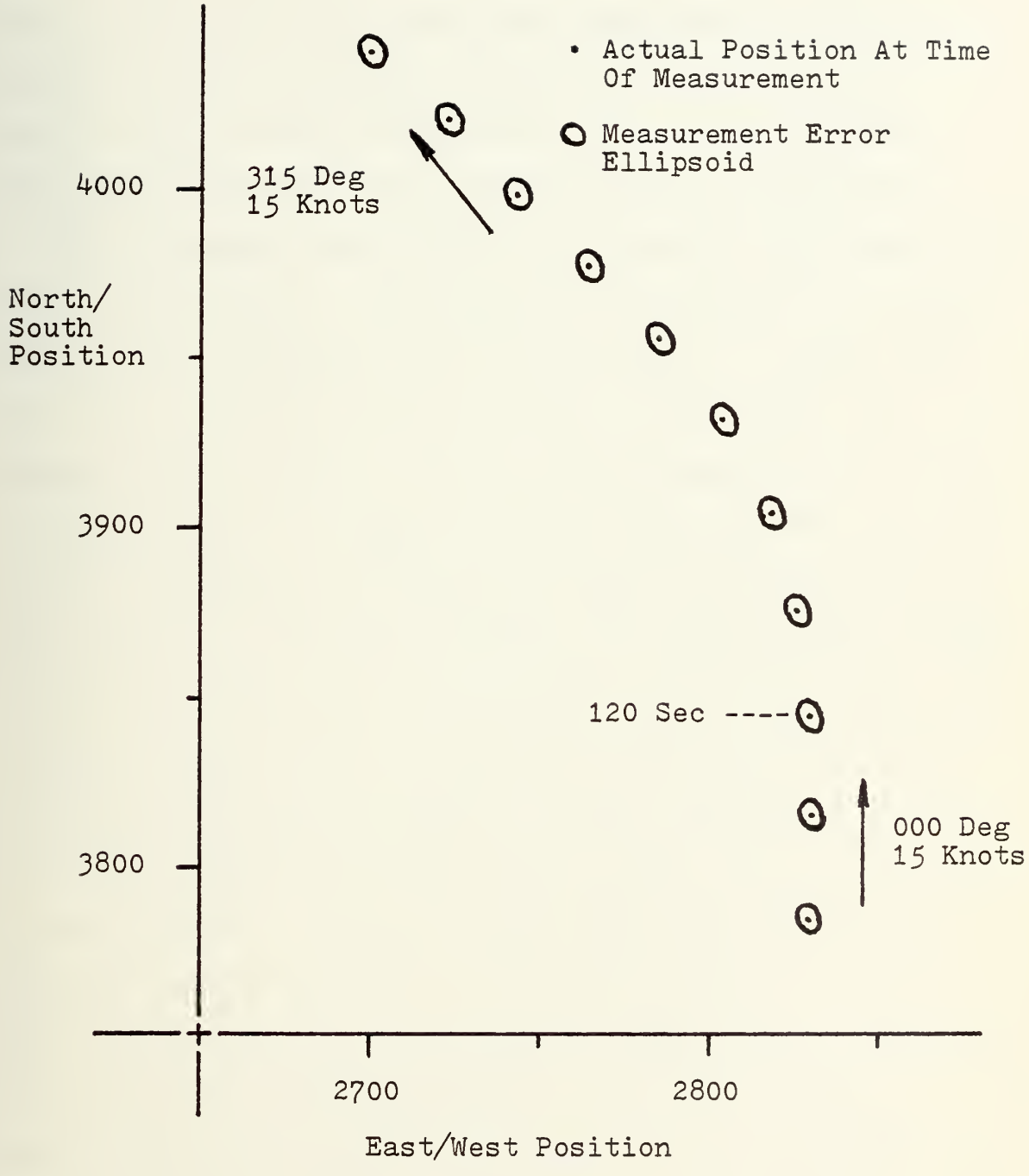


FIGURE 6 SAMPLED GEOGRAPHIC TRACK FOR TARGET 3 DURING 45 DEGREE COURSE CHANGE

nonmaneuvering and the maneuvering target scenarios are listed in Table II. There was good correlation between the measured variances of error and the computed variances. Additionally, the mean errors were approximately equal to minus one-half the values of the least significant bit used in the interface simulation. This is in agreement with the theoretical mean error generated by truncation.

Each scenario was repeated for seventeen logarithmically distributed values of the Kalman filter variance of target acceleration parameter from 0.0005 yd^2/sec^4 (0.00042 m^2/sec^4) to 1 yd^2/sec^4 (0.85 m^2/sec^4). In all cases, the Kalman filter achieved a steady state condition in 60 sec, corresponding to 17 target position measurements. Steady state is defined as the condition where the Kalman filter gain matrix (K_n) would be constant for targets with constant relative position and constant sample intervals. The bearing, range, course and speed of each target was computed from the Kalman filter's optimal estimate of target state and compared with the actual target bearing, range, course and speed. Error averages were computed for the time interval from 60 sec to 300 sec. Figures 7, 8, 9 and 10 show plots of the average root mean square of error for the target bearings, ranges, courses and speeds, respectively, as a function of the variance of target acceleration parameter. It should be noted that the error averages for the maneuvering target are a function of the elapsed time between maneuvers and the time spent in

	NONMANEUVERING TARGET	MANEUVERING TARGET
Bearing Error		
Mean:	-0.01406 deg	-0.01893 deg
Variance:	0.01514 deg^2	0.01580 deg^2
Std. Dev.:	0.1231 deg	0.1257 deg
Range Error		
Mean:	-2.603 yds (2.381 m)	-2.175 yds (1.989 m)
Variance:	51.53 yds^2 (43.08 m^2)	48.28 yds^2 (40.37 m^2)
Std. Dev.:	7.178 yds (6.564 m)	6.948 yds (6.354 m)

TABLE II MEASUREMENT ERROR STATISTICS

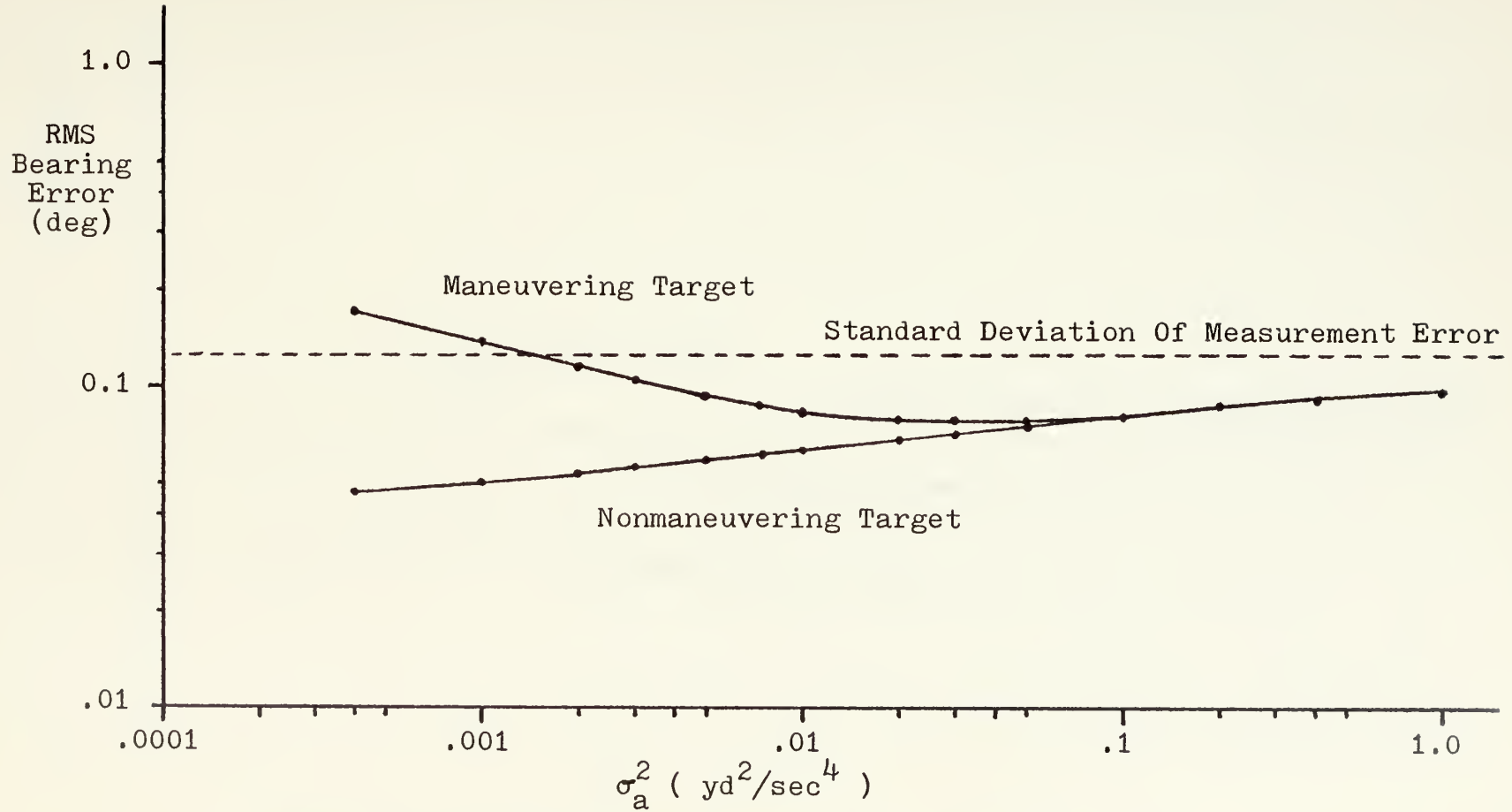


FIGURE 7 AVERAGE ROOT MEAN SQUARE OF BEARING ERROR VS. VARIANCE OF TARGET ACCELERATION PARAMETER (σ_a^2)

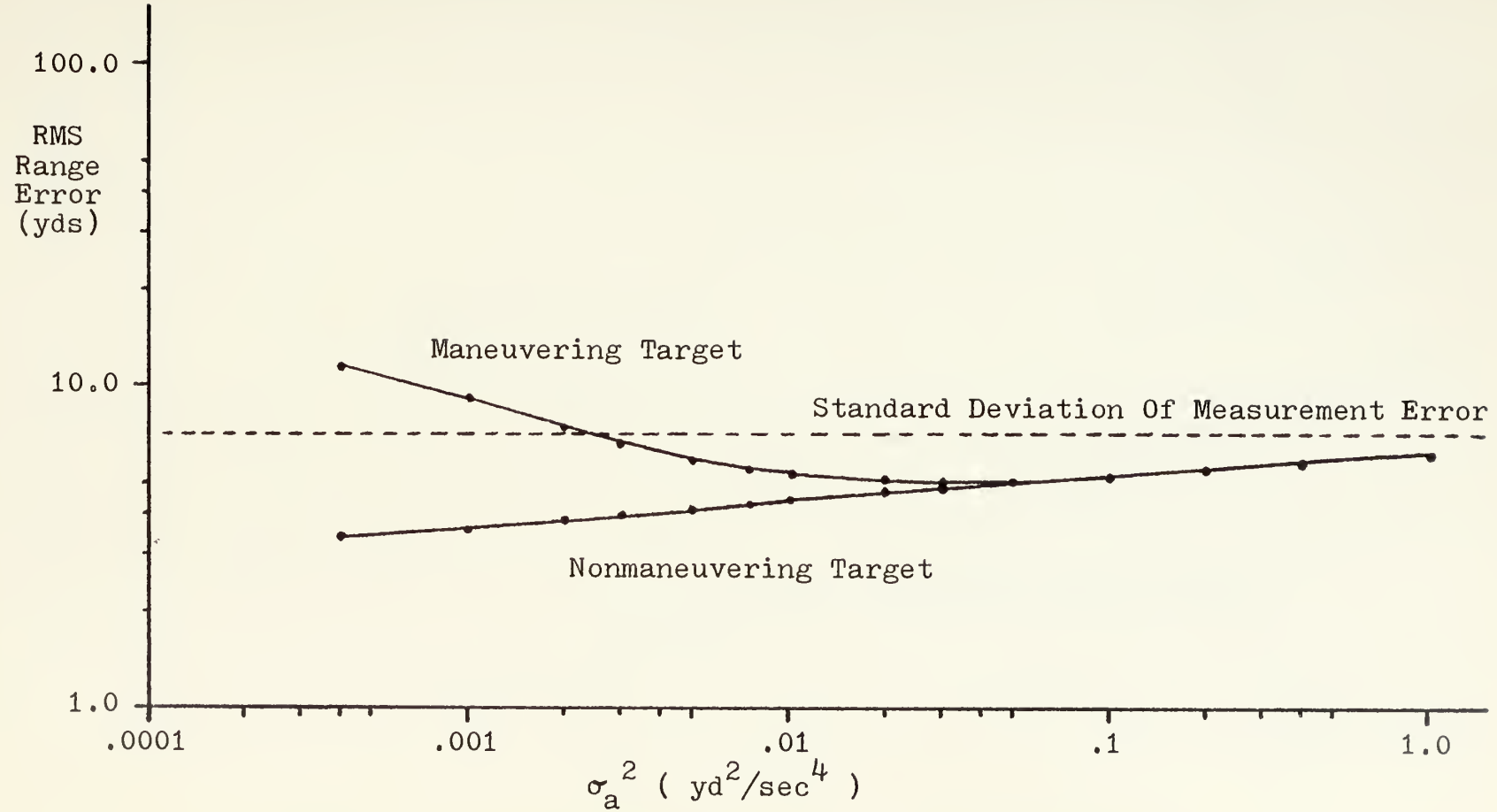


FIGURE 8 AVERAGE ROOT MEAN SQUARE OF RANGE ERROR VS. VARIANCE OF TARGET ACCELERATION PARAMETER (σ_a^2)

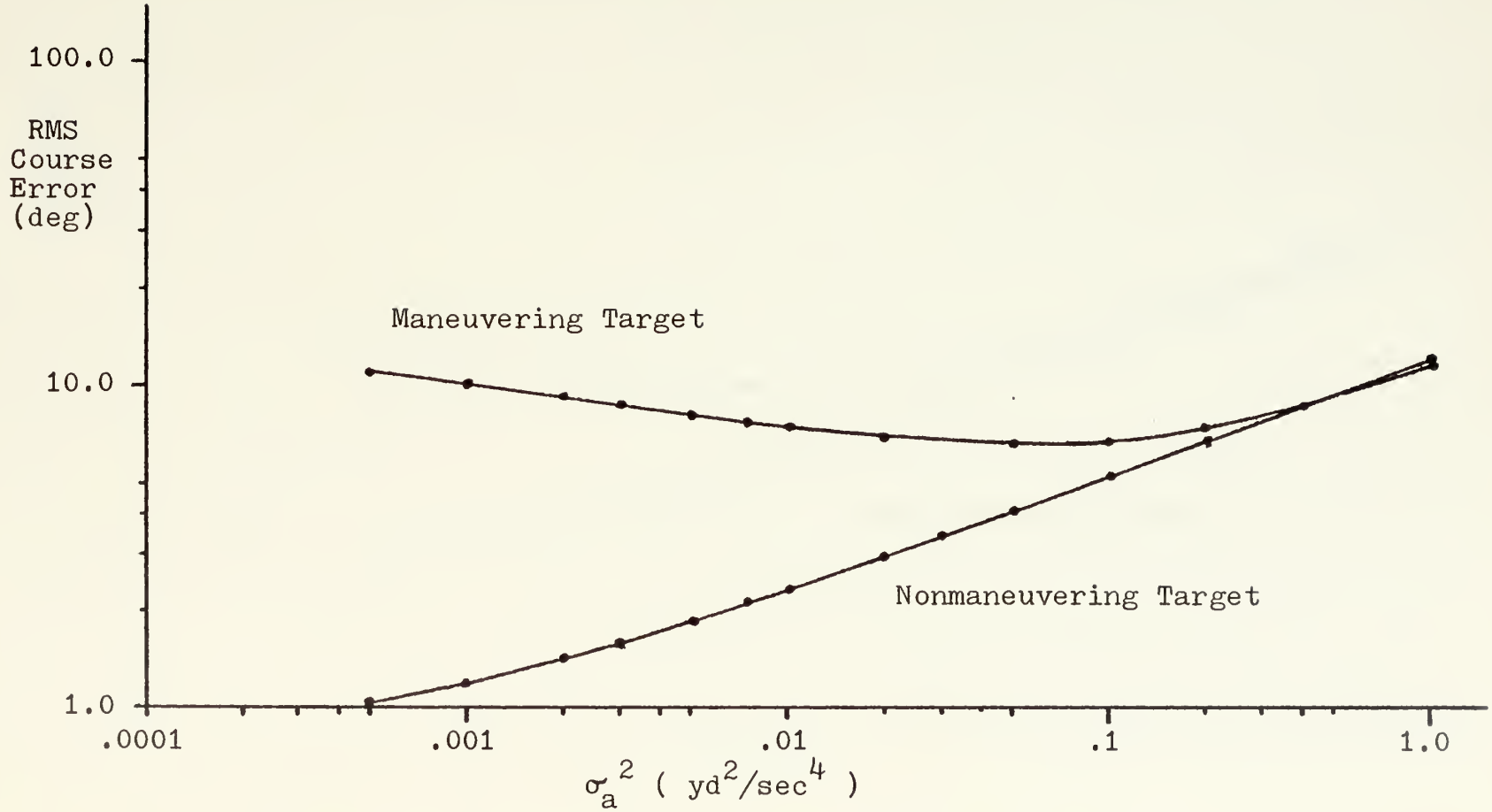


FIGURE 9 AVERAGE ROOT MEAN SQUARE OF COURSE ERROR VS. VARIANCE OF TARGET ACCELERATION PARAMETER (σ_a^2)

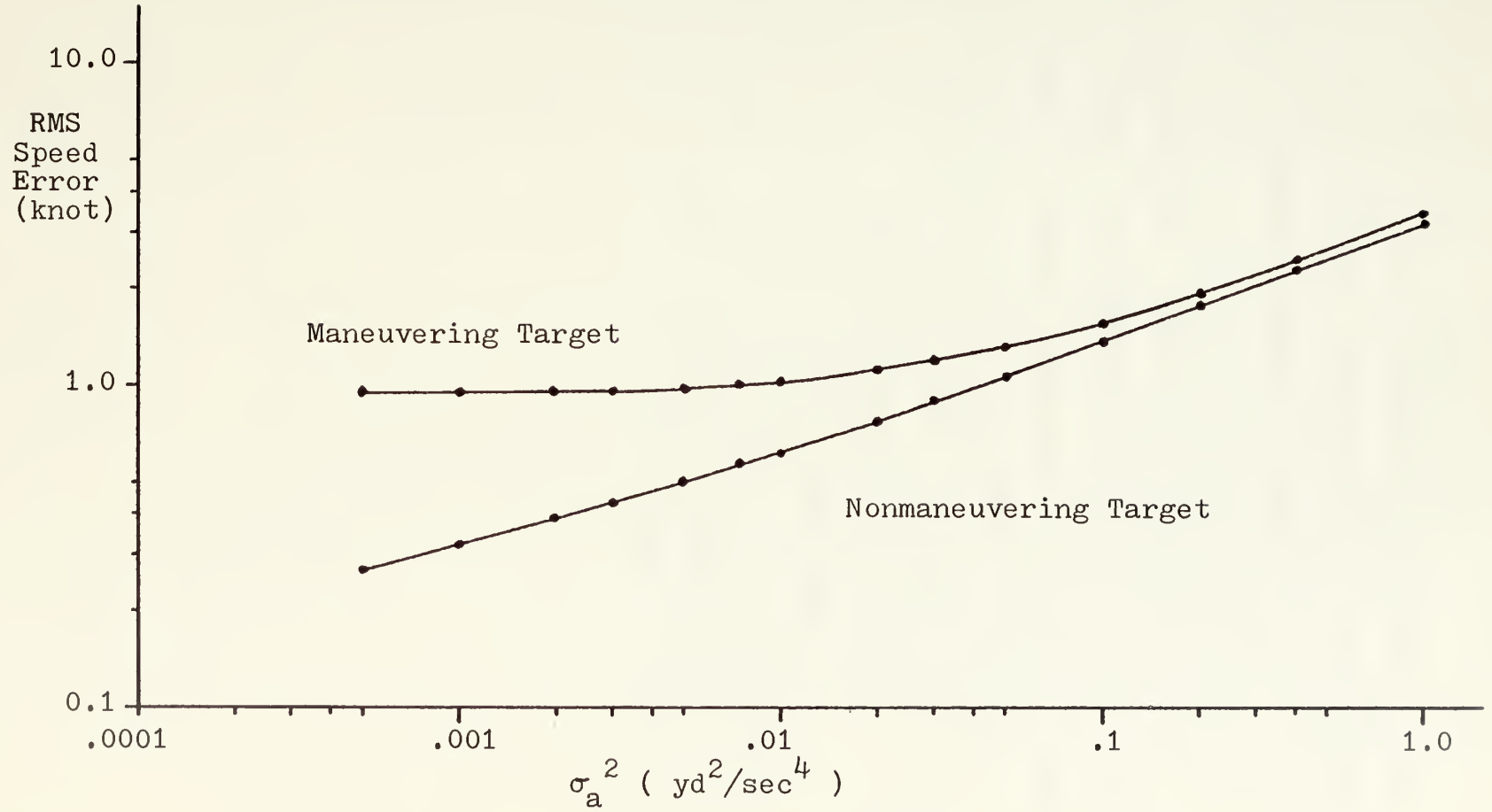


FIGURE 10 AVERAGE ROOT MEAN SQUARE OF SPEED ERROR VS. VARIANCE OF TARGET ACCELERATION PARAMETER (σ_a^2)

maneuver. Thus, the plots for the maneuvering target only apply for the particular scenario used.

Several conclusions can be drawn from these figures. For nonmaneuvering targets, the variance of acceleration parameter (σ_a^2) should be small ($< .001$ yd^2/sec^4 ($.0008$ m^2/sec^4)) if smooth course and speed values are desired. For maneuvering targets, there is an optiaml value of σ_a^2 ($\approx .10$ yd^2/sec^4 ($.08$ m^2/sec^4)) below which the filter does not follow the target maneuvers well and above which the target data will be noisy regardless of the target maneuvers. There is a lower limit to how well the Kalman filter will track a maneuvering target even with an optimal value of σ_a^2. Finally, the bearing and range to a non-maneuvering target will be improved by the Kalman filter, as shown by the standard deviation of measurement error line on Figures 7 and 8, over a wide range of σ_a^2 values.

V. MICROPROCESSOR PROGRAM

A MCS-8 microcomputer system, using INTEL's 8080B microprocessor, was used to implement the Kalman filter algorithm. The objective of implementing the algorithm was to determine limits to the time required to accomplish the Kalman filter computations and the accuracy of the results.

A. PROGRAMMING LANGUAGE

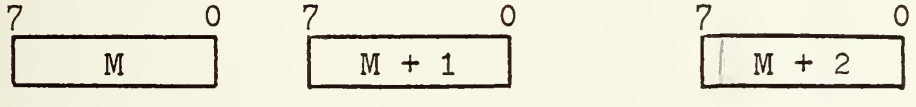
INTEL Corporation has developed the PL/M compiler, a derivative of PL/1, for use with their 8080 microprocessor. PL/M is a block structured language which speeds microprocessor program development by relieving the programmer of the tedious task of memory and register management [1]. It facilitates the use of variable names while still permitting the programmer the freedom to reference specific memory locations. The basic instruction set includes arithmetic and Boolean algebra operation on 8-bit and 16-bit variables and/or constants. Program control is accomplished with Procedure, Do, Go To, If, Call and Return statements. Logic functions such as comparison, shift and rotation are also included. Floating point and transcendental functions must be implemented by subroutines. PL/M permits the program control provided by assembly languages while taking a much more readable form.

B. FLOATING POINT ARITHMETIC

Floating point arithmetic is required whenever the range of numbers represented by the program variables is large or unpredictable [12]. Such is the case with the Kalman filter algorithm. This is particularly true if the operator is given the freedom to alter the filter parameters. A disadvantage of using floating point arithmetic is the additional execution time required.

A suitable set of floating point arithmetic procedures were not available. The procedures listed in Appendix B were derived from several existing procedures and the general guidelines detailed in Reference 12.

A 16-bit mantissa, 7-bit exponent and 1-bit sign were chosen to represent the floating point numbers using three 8-bit data words in the following format:



Mantissa (16-bits)

Sign (1-bit)
Exponent (7-bits)

M - Memory Address Associated with the Floating Point Number

The 16-bit mantissa was chosen because it is the largest binary number that INTEL's 8080 microprocessor can manipulate without a considerable increase in the complexity of the programming. Additionally, it is compatible with the input and output data. The mantissa is always left justified with the binary point located to the left of the most

significant bit. Thus the mantissa is always equal to or greater than 0.5, and less than 1.0.

The sign was located in the third word in order that the mantissa be a full 16-bits. A positive sign is represented by a binary 0 and a negative sign is represented by a binary 1.

The remaining 7-bits of the third word are used for the exponent. The value of the exponent is equal to 64 plus the power of two to be associated with the mantissa.

The data and constants in the PL/M programs are written in this format with hexadecimal numbers. The following are examples of this form.

DECIMAL	HEXADECIMAL FLOATING POINT
0.5	80H, 00H, 40H
1.0	80H, 00H, 41H
-1.0	80H, 00H, C1H
10.0	A0H, 00H, 44H
64.0	80H, 00H, 47H
0.1	CCH, CDH, 3DH

The largest magnitude number that can be represented is FFH, FFH, 7FH, which corresponds to 9.223×10^{18}. The smallest magnitude number that can be represented is 80H, 00H, 00H which corresponds to 2.710×10^{-20}. Zero is represented by 00H, 00H, 00H.

The typical execution time of the arithmetic and trigonometric procedures listed in Appendix B, using INTEL's

8080B microprocessor, are listed below.

FLOATING POINT PROCEDURE	MAXIMUM EXECUTION TIME (msec)
Addition (ADD)	1.3
Subtraction (SUB)	1.4
Multiplication (MULT)	3.4
Division (DIV)	4.1
Comparison (COMPARE)	0.8
Square Root (SQRT)	17.0
Cos (TRIG 1)	18.0
Sin (TRIG 2)	18.0
Cos and Sin (TRIG 3)	30.0
Arctangent (TRIG 4)	19.0

The actual execution times of these procedures are a function of the numbers involved.

Overflow and underflow provisions are included in the arithmetic procedures. Overflow will result in a solution equal to the largest possible number with the appropriate sign. Underflow will result in a solution equal to zero.

The precision of the sine and cosine functions, using the procedure "TRIG", is at worst $\pm$ 0.000488 if the argument is in the range 0 to 2π. The precision of the arctangent function, using the procedure "TRIG", is equal to $\pm$ 0.0610 mrad.

The four arithmetic procedures are used by passing the variables' addresses to the procedure with a call statement. The following PL/M statements are examples.

FUNCTION	PL/M STATEMENTS
X + Y = Z	Call ADD (.X, .Y, .Z);
X - Y = Z	Call SUB (.X, .Y, .Z);
A x B = C	Call MULT (.A, .B, .C);
L ÷ M = Q	Call DIV (.L, .M, .Q);

Note: The dot symbol proceeding a variable name in PL/M indicates the address of the variable. Some variables may correspond to addresses of other variables, in which case the dot would be omitted.

The "COMPARE" procedure returns a binary integer quantity which indicates the relationship of the arguments.

PL/M STATEMENT

```
N = COMPARE (.A, .B);
```

RESULT

```
If A < B     Then N = 0
If A = B     Then N = 1
If A > B     Then N = 2
```

The argument for "SQRT" must be positive.

FUNCTION	PL/M STATEMENT
$Y = \sqrt{X}$	Call SQRT (.X, .Y);

The procedure "TRIG" can be used to compute the sine and/or cosine, and arctangent by passing the addresses of the arguments plus a function indicator flag to the procedure with a call statement. It requires less time to

compute sine and cosine of the same angle with one call rather than two. The following are examples of the multiple uses of the "TRIG" procedure.

FUNCTION	PL/M STATEMENTS
X = Cos (A)	Call TRIG (.X, .W, .A, 1);
Y = Sin (A)	Call TRIG (.W, .Y, .A, 2);
X = Cos (A) and Y = Sin (A)	Call TRIG (.X, .Y, .A, 3);
A = Arctan (Y/X)	Call TRIG (.X, .Y, .A, 4);

Note: W is a dummy variable used to fill the unused argument location in the call statement when only cosine or sine is desired.

C. TRACKING SYSTEM PROGRAM

An experimental version of the tracking system software was written to determine the execution time of the Kalman filter algorithm and the error statistics of the resulting target track data. This program, written in PL/M, is listed in Appendix C. There are three functional parts to this program; (1) data input, (2) the Kalman filter, and (3) computation of output data.

The data input portion of the program includes the simulated interface system output data, and the simulated control panel functions. The simulated interface system data was generated in the correct floating point format by the FORTRAN program used to model the system performance. The variables "BRG," "RNG," and "TIM" correspond to the

measured target bearing, range and time of detection, respectively. The own ship north/south and east/west velocity are represented by the constants "NSVEL" and "EWVEL." The Kalman filter parameters σ_a^2, σ_θ^2, and σ_r^2 are represented by "SIGA," "SIGTH," and "SIGR." The variables "MARKOLD" and "CONTROL" serve the functions of distinguishing which target data corresponds to existing tracks, and which control panel switches are turned on. These variables are associated with the variables "SIMFLAG ( )" and "DFLAG" which correspond to the interface unit from which the target data came and thus the target's track number. In an actual operating system the program would use INPUT statements or direct memory access to read in the target measurement and control panel switch data. The measurement data would have to be converted from fixed point to floating point format before processing.

The Kalman filter statements are grouped into two long DO statements. The first is used to initialize the filter when a new target is first processed. The second group is used to process new data for existing target tracks. The sequence of statements and variable names parallel the algebraic expressions listed in Appendix A. Temporary variables were introduced where necessary to avoid repetitive operations and to store intermediate results.

Using the Kalman filter optimal estimates of target position and relative velocity, the third portion of the program computes and displays the target bearing, range,

course and speed.

Following the program debugging, a series of tests, using a full range of typical input data, were run to verify the correct computation of each variable. The FORTRAN model was used to compute the "correct" values for each variable using the Naval Postgraduate School IBM 360 computer. It was found that the computational errors were confined to a range of significance sufficiently small to avoid instabilities in the filter. After processing several measurements, the Kalman filter estimates of target state computed with the microprocessor were within the same range of error as the IBM 360 solutions. The covariance of estimation error and Kalman gain matrix elements computed by the microprocessor converged rapidly on the values computed by the IBM 360 computer.

As a check on the microprocessor's performance a series of eight test runs were made, each with a different target. The eight simulated targets were identical to the non-maneuvering targets used previously except the signs of the east/west initial positions for targets 3-8 were negative. This change was made because the execution time and precision of the "TRIG" procedure is worst for targets in the north-west quadrant. Nonmaneuvering targets were used because the available memory space limited the amount of data that could be loaded with the program. The test runs were restricted to 130 sec tracks. A variance of acceleration of .001 yd^2/sec^4 (.00084 m^2/sec^4) was selected

to insure that computational errors would not be overshadowed by normal estimation error in the output. It was assumed that these tests would represent worst case conditions.

The test program occupied a total of $2E00_{16}$ bytes of memory. This was divided as follows:

Floating Point Procedures	$98A_{16}$ Bytes
Kalman Filter Algorithm	$12C7_{16}$ Bytes
Bearing, Range, Course and Speed Computations	220_{16} Bytes
Variable Storage	363_{16} Bytes

The error averages for the test data are listed in Table III. These error averages represent the performance of the microprocessor tracking program after the Kalman filter achieved a steady state condition, approximately 50 sec after initialization. This resulted in a relatively small sample set of 20-21 samples for each target. The error statistics of the input data during this time are listed in Table IV. As compared to the measurement error statistics presented in Table II, there is less correlation with the theoretical measurement errors for the microprocessor test data. Therefore, the error averages listed in Table III should be considered typical only for a short period of operation rather than for long periods with all possible targets.

An analysis of these results reveals the microprocessor's influence on the track data. By comparing the course and

TARGET NO.	BEARING (degrees) Mean	RMS	RANGE (yards) Mean	RMS	COURSE (degrees) Mean	RMS	SPEED (knots) Mean	RMS
1	-0.032	0.074	0.31	1.92	0.145	0.628	-0.016	0.164
2	-0.018	0.058	-0.46	2.59	-0.054	0.329	-0.139	0.279
3	0.015	0.167	-2.10	2.50	-0.083	0.744	-0.024	0.192
4	0.059	0.205	-1.17	1.92	0.002	0.337	0.149	0.177
5	-0.017	0.027	1.08	2.16	-0.204	0.952	-0.046	0.224
6	-0.025	0.030	-1.10	3.55	-0.174	0.638	-0.006	0.368
7	-0.050	0.031	-1.11	4.34	-3.459	1.998	-1.247	0.253
8	0.028	0.001	-0.99	3.84	0.492	0.771	0.764	0.790

TABLE III MICROPROCESSOR TRACKING PROGRAM ERROR AVERAGES

TARGET NO.	BEARING (degrees) Mean	Std. Dev.	RANGE (yards) Mean	Std. Dev.
1	0.0106	0.1127	-3.633	5.634
2	0.0002	0.1446	-1.169	6.033
3	-0.0247	0.1188	-0.702	5.014
4	0.0070	0.1357	-1.693	6.788
5	-0.0261	0.1057	-0.424	8.460
6	-0.0075	0.1105	-2.619	8.351
7	-0.0035	0.1072	0.810	4.536
8	-0.0564	0.1452	-1.872	6.805

TABLE IV MEASUREMENT ERROR STATISTICS FOR MICROPROCESSOR TRACKING PROGRAM INPUT DATA

speed data for targets 3, 5 and 7 with the data for targets 4, 6 and 8 it can be seen that for targets with similar course and speed the errors increase with range. Additionally, errors are larger for targets with small relative velocities. The combination of long range and small relative velocities can be expected to result in larger errors due to the limitations of the floating point arithmetic to handle large numbers with small differences.

The final parameter of interest is the execution time of the microprocessor program. The INTEL 8080B microprocessor was able to execute 85 consecutive iterations of the Kalman filter algorithm in 37.5 secs, resulting in an average execution time of 0.44 secs. The computation of target course and speed required a worst case execution time of 54 msec. It would not be necessary to compute the target course and speed for every target on each antenna scan. The additional processing required to output the course and speed data would best be accomplished by a separate microprocessor dedicated to each display device. In general, a network of microprocessors could be used to share the task and thus increase the number of targets which could be tracked.

VI. CONCLUSION

The tedious, error-prone task of tracking surface search radar targets manually, as is common onboard naval ships not equipped with NTDS, can be eliminated with the use of large scale integrated microprocessors.

The AN/SPS-10 Surface Search Radar, installed on most U.S. Navy ships, could provide target position measurements with sufficient accuracy to perform automatic tracking of surface radar targets. Using leading edge detection and currently available hardware it would be possible to digitally measure the bearing and range to a target with a standard deviation of error of 0.126 deg and 7.15 yards (6.54 m) respectively.

Using a 4-state Kalman filter and appropriate conversions for polar coordinate measurements, it is possible to obtain optimal estimates of target north/south and east/west position and velocity. From this data and knowledge of own ship's motion, target course and speed can be computed far more accurately than by manual tracking techniques.

A set of floating point arithmetic procedures was developed for INTEL's 8080 microprocessor based on a floating point number format with a 16-bit mantissa and a 7-bit exponent. These procedures maintain sufficient accuracy to reliably compute the Kalman filter variables

while limiting the execution time sufficiently to permit complex functions, like the Kalman filter, to be implemented.

The Kalman filter, and course and speed calculations were successfully implemented using the microprocessor development system at the Naval Postgraduate School. The average execution time of the Kalman filter was 0.44 secs while the course and speed calculations required 54 msec. This was accomplished with computed course and speed errors for nonmaneuvering targets typically less than 2.0 deg and 0.5 knots. With the 3.5 sec sweep of the AN/SPS-10, a single system could continuously provide course and speed data for seven targets. This could be increased by using faster microprocessors currently entering the market.

It is recommended that future work in this area be devoted to (1) improvements in the floating-point software, (2) development of the interface hardware, and (3) improvements to the Kalman filter algorithm. The floating point software could be improved by reducing the execution time of the arithmetic procedures and increasing the precision of the transcendental functions. The Kalman filter algorithm could be improved by minimizing the effects of own ship's motion, and reducing the execution time.

A microprocessor-based tracking system has the potential to provide any ship with a digital automatic surface tracking capability, without the expense of NTDS size equipment. If implemented, it could reduce the manning of underway watches while improving the quality of the tactical

information available to the Officer of the Deck. There are additional applications of similar systems for such functions as sonar tracking and navigation.

1. $\tilde{P}(n) = \Phi \hat{P}(n-1)\Phi^T + \Gamma Q \Gamma^T$

$\tilde{P}_{11}(n) = \hat{P}_{11}(n-1) + (\hat{P}_{21}(n-1) + \hat{P}_{12}(n-1)\ ) T + \hat{P}_{22}(n-1)\ T^2 + \sigma_a^2\ T^4/4$

$\tilde{P}_{12}(n) = \hat{P}_{12}(n-1) + \hat{P}_{22}(n-1)\ T + \sigma_a^2\ T^3/2$

$\tilde{P}_{13}(n) = \hat{P}_{13}(n-1) + (\hat{P}_{23}(n-1) + \hat{P}_{14}(n-1)\ ) T + \hat{P}_{24}(n-1)\ T^2$

$\tilde{P}_{14}(n) = \hat{P}_{14}(n-1) + \hat{P}_{24}(n-1)\ T$

$\tilde{P}_{21}(n) = \hat{P}_{21}(n-1) + \hat{P}_{22}(n-1)\ T + \sigma_a^2\ T^3/2$

$\tilde{P}_{22}(n) = \hat{P}_{22}(n-1) + \sigma_a^2\ T^2$

$\tilde{P}_{23}(n) = \hat{P}_{23}(n-1) + \hat{P}_{24}(n-1)\ T$

$\tilde{P}_{24}(n) = \hat{P}_{24}(n-1)$

$$\tilde{P}_{31}(n) = \hat{P}_{31}(n-1) + (\hat{P}_{41}(n-1) + \hat{P}_{32}(n-1)\ )\ T + \hat{P}_{42}(n-1)\ T^2$$

$$\tilde{P}_{32}(n) = \hat{P}_{32}(n-1) + \hat{P}_{42}(n-1)\ T$$

$$\tilde{P}_{33}(n) = \hat{P}_{33}(n-1) + (\hat{P}_{43}(n-1) + \hat{P}_{34}(n-1)\ )\ T + \hat{P}_{44}(n-1)\ T^2 + \sigma_a^2\ T^4/4$$

$$\tilde{P}_{34}(n) = \hat{P}_{34}(n-1) + \hat{P}_{44}(n-1)\ T + \sigma_a^2\ T^3/2$$

$$\tilde{P}_{41}(n) = \hat{P}_{41}(n-1) + \hat{P}_{42}(n-1)\ T$$

$$\tilde{P}_{42}(n) = \hat{P}_{42}(n-1)$$

$$\tilde{P}_{43}(n) = \hat{P}_{43}(n-1) + \hat{P}_{44}(n-1)\ T + \sigma_a^2\ T^3/2$$

$$\tilde{P}_{44}(n) = \hat{P}_{44}(n-1) + \sigma_a^2\ T^2$$

2. $$R(n) = \begin{vmatrix} \sigma_x^2(n) & \sigma_{xy}^2(n) \\ \sigma_{xy}^2(n) & \sigma_y^2(n) \end{vmatrix}$$

1. $$\sigma_x^2(n) = \sigma_r^2 \sin^2 \theta(n) + r^2(n)\, \sigma_\theta^2 \cos^2 \theta\,(n)$$

2. $$\sigma_y^2(n) = \sigma_r^2 \cos^2 \theta(n) + r^2(n)\, \sigma_\theta^2 \sin^2 \theta\,(n)$$

3. $$\sigma_{xy}^2(n) = \tfrac{1}{2} \sin 2\,\theta(n) \left[\sigma_r^2 - r^2(n)\, \sigma_\theta^2 \right]$$

3. $K(n) = \tilde{P}(n)\, H^T (H\, \tilde{P}(n)\, H^T + R(n))^{-1}$

$$\Delta = (\tilde{P}_{11}(n) + \sigma_x^2(n))(\tilde{P}_{33}(n) + \sigma_y^2(n)) - (\tilde{P}_{31}(n) + \sigma_{xy}^2(n))(\tilde{P}_{13}(n) + \sigma_{xy}^2(n))$$

$$Axx(n) = \tilde{P}_{11}(n)(\tilde{P}_{33}(n) + \sigma_y^2(n))/\Delta - \tilde{P}_{13}(n)(\tilde{P}_{31}(n) + \sigma_{xy}^2(n))/\Delta$$

$$Axy(n) = \tilde{P}_{13}(n)(\tilde{P}_{11}(n) + \sigma_x^2(n))/\Delta - \tilde{P}_{11}(n)(\tilde{P}_{13}(n) + \sigma_{xy}^2(n))/\Delta$$

$$\underline{Bxx(n)} = \tilde{P}_{21}(n)(\tilde{P}_{33}(n) + \sigma_y^2(n))/\Delta - \tilde{P}_{23}(n)(\tilde{P}_{31}(n) + \sigma_{xy}^2(n))/\Delta$$
$$T$$

$$\underline{Bxy(n)} = \tilde{P}_{23}(n)(\tilde{P}_{11}(n) + \sigma_x^2(n))/\Delta - \tilde{P}_{21}(n)(\tilde{P}_{13}(n) + \sigma_{xy}^2(n))/\Delta$$
$$T$$

$$Ayx(n) = \tilde{P}_{31}(n)(\tilde{P}_{33}(n) + \sigma_y^2(n))/\Delta - \tilde{P}_{33}(n)(\tilde{P}_{31}(n) + \sigma_{xy}^2(n))/\Delta$$

$$Ayy(n) = \tilde{P}_{33}(n)(\tilde{P}_{11}(n) + \sigma_x^2(n))/\Delta - \tilde{P}_{31}(n)(\tilde{P}_{13}(n) + \sigma_{xy}^2(n))/\Delta$$

$$\underline{Byx(n)} = \tilde{P}_{41}(n)(\tilde{P}_{33}(n) + \sigma_y^2(n))/\Delta - \tilde{P}_{43}(n)(\tilde{P}_{31}(n) + \sigma_{xy}^2(n))/\Delta$$
$$T$$

$$\underline{Byy(n)} = \tilde{P}_{43}(n)(\tilde{P}_{11}(n) + \sigma_x^2(n))/\Delta - \tilde{P}_{41}(n)(\tilde{P}_{13}(n) + \sigma_{xy}^2(n))/\Delta$$
$$T$$

4. $\hat{P}(n) = (I - K(n)\, H)\, \tilde{P}(n)$

$\hat{P}_{11}(n) = \tilde{P}_{11}(n)\ (1 - Axx) - \tilde{P}_{31}(n)\ Axy$

$\hat{P}_{12}(n) = \tilde{P}_{12}(n)\ (1 - Axx) - \tilde{P}_{32}(n)\ Axy$

$\hat{P}_{13}(n) = \tilde{P}_{13}(n)\ (1 - Axx) - \tilde{P}_{33}(n)\ Axy$

$\hat{P}_{14}(n) = \tilde{P}_{14}(n)\ (1 - Axx) - \tilde{P}_{34}(n)\ Axy$

$\hat{P}_{21}(n) = \tilde{P}_{21}(n) - \tilde{P}_{11}(n)\ Bxx - \tilde{P}_{31}(n)\ Bxy$

$\hat{P}_{22}(n) = \tilde{P}_{22}(n) - \tilde{P}_{12}(n)\ Bxx - \tilde{P}_{32}(n)\ Bxy$

$\hat{P}_{23}(n) = \tilde{P}_{23}(n) - \tilde{P}_{13}(n)\ Bxx - \tilde{P}_{33}(n)\ Bxy$

$\hat{P}_{24}(n) = \tilde{P}_{24}(n) - \tilde{P}_{14}(n)\ Bxy - \tilde{P}_{34}(n)\ Bxy$

$$\hat{P}_{31}(n) = \tilde{P}_{31}(n)\ (1 - Ayy) - \tilde{P}_{11}(n)\ Ayx$$

$$\hat{P}_{32}(n) = \tilde{P}_{32}(n)\ (1 - Ayy) - \tilde{P}_{12}(n)\ Ayx$$

$$\hat{P}_{33}(n) = \tilde{P}_{33}(n)\ (1 - Ayy) - \tilde{P}_{13}(n)\ Ayx$$

$$\hat{P}_{34}(n) = \tilde{P}_{34}(n)\ (1 - Ayy) - \tilde{P}_{14}(n)\ Ayx$$

$$\hat{P}_{41}(n) = \tilde{P}_{41}(n) - \tilde{P}_{11}(n)\ Byx - \tilde{P}_{31}(n)\ Byy$$

$$\hat{P}_{42}(n) = \tilde{P}_{42}(n) - \tilde{P}_{12}(n)\ Byx - \tilde{P}_{32}(n)\ Byy$$

$$\hat{P}_{43}(n) = \tilde{P}_{43}(n) - \tilde{P}_{13}(n)\ Byx - \tilde{P}_{33}(n)\ Byy$$

$$\hat{P}_{44}(n) = \tilde{P}_{44}(n) - \tilde{P}_{14}(n)\ Byx - \tilde{P}_{34}(n)\ Byy$$

```
100H:

DECLARE ZE BYTE, ZZ ADDRESS;
DECLARE YE BYTE, XE BYTE;

  /*                                  */
  /* FLOATING POINT ADD ROUTINE       */
  /*                                  */
ADD: PROCEDURE (XA,YA,ZA);
        DECLARE (XA,YA,ZA,XX,YY) ADDRESS,
                (I,XE,YE,RANGE,SIGNSEQUAL) BYTE,
                X BASED XA BYTE,
                Y BASED YA BYTE,
                Z BASED ZA BYTE;

  /*                                                  */
  /* PROCEDURE TO LEFT JUSTIFY MANTISSA IN BINARY     */
  /*                                                  */
    ADJUST: PROCEDURE;
        DO I=0 TO 15;
            IF HIGH(ZZ)>07FH THEN RETURN;
            IF (ZE AND 7FH)=0H THEN DO;
                    ZE=0H; . ZZ=0H;  RETURN;
                END;
            ZE = ZE-1;
            ZZ = SHL(ZZ,1);
        END;
    END ADJUST;

  /* DETERMINE DIFFERENCE IN EXPONENTS */
    IF (XE:=X(2) AND 7FH)>(YE:=Y(2) AND 7FH) THEN RANGE=XE-YE;
        ELSE RANGE=YE-XE;

  /* CHECK TO SEE IF NUMBERS ARE WITHIN SIGNIFICANCE RANGE */
    IF RANGE>15 THEN DO;

  /* VARIABLES NOT WITHIN SIGNIFICANCE RANGE */
        IF XE>YE THEN DO;
            Z=X; Z(1)=X(1); Z(2)=X(2); RETURN;
            END;
        Z=Y; Z(1)=Y(1); Z(2)=Y(2); RETURN;
        END;
```

```
/* VARIABLES ARE WITHIN RANGE OF SIGNIFICANCE */

/* FORM MANTISSAS */
  XX=SHL(DOUBLE(X),8) OR X(1);
  YY=SHL(DOUBLE(Y),8) OR Y(1);
  IF (X(2) AND 80H)=(Y(2) AND 80H) THEN SIGNSECLAL=1;
       ELSE SIGNSEQUAL=0;


/* EXPONENTS EQUAL */
  IF YE=XE THEN DO;

/* Y>X */
       IF YY>XX THEN DO;
            ZE= Y(2);
            IF SIGNSEQUAL THEN GO TO EXIT1;
            GO TO EXIT2;
       END;

/* X>Y */
       IF YY<XX THEN DO;
            ZE =X(2);
            IF SIGNSEQUAL THEN GO TO EXIT1;
            GO TO EXIT3;
       END;

/* X=Y */
       IF SIGNSEQUAL THEN DO;
            Z=X; Z(1)=X(1); Z(2)=X(2)+1;
            RETURN;
            END;
       Z=0; Z(1)=0; Z(2)=0;
       RETURN;
       END;


/* EXPONENT OF X > EXPONENT OF Y */
  IF XE>YE THEN DO;
       ZE=X(2);
       YY=SHR( YY,RANGE );
       IF SIGNSEQUAL THEN GO TO EXIT1;
       GO TO EXIT3;
       END;
```

```
      /* EXPONENT OF Y > EXPONENT OF X */
               ZE=  Y(2);
               XX=SHR(XX,RANGE);
               IF SIGNSEQUAL THEN GO TO EXIT1;
               GO TC EXIT2;

      /* WHEN SIGNS OF THE MANTISSAS ARE EQUAL THE */
      /* NUMBERS ARE ADDED                         */
   EXIT1:
      ZZ=XX+YY;
      IF CARRY THEN DO;
           ZZ=SCR(ZZ,1);
           ZE=ZE  +1;
            END;
            Z=HIGH(ZZ); Z(1)=LOW(ZZ); Z(2)=ZE;
            RETURN;

      /* WHEN SIGNS ARE DIFFERENT AND Y>X */
   EXIT2:
               ZZ=YY-XX;
               CALL ADJUST;
               Z=HIGH(ZZ); Z(1)=LOW(ZZ); Z(2)=ZE;
               RETURN;

      /* WHEN SIGNS ARE DIFFERENT AND X>Y */
   EXIT3:
               ZZ=XX-YY;
               CALL ADJUST;
               Z=HIGH(ZZ); Z(1)=LOW(ZZ); Z(2)=ZE;
               RETURN;
   END ADD;
```

```
   /*                                                    */
   /* FLOATING POINT SUBTRACTION ROUTINE                 */
   /*                                                    */
SUB: PROCEDURE (XA,YA,ZA);
     DECLARE (XA,YA,ZA) ADDRESS,
          YY BASED YA BYTE,
          YYMINUS (3) BYTE;

     YYMINUS=YY;
     YYMINUS(1)=YY(1);
     YYMINUS(2)=YY(2) XOR 80H;
     CALL ADD(XA,.YYMINUS,ZA);
END SUB;
```

```
  /*                                                    */
  /* FLOATING POINT MULTIPLY ROUTINE                    */
  /*                                                    */
MULT: PROCEDURE(XA,YA,ZA);
    DECLARE (XA,YA,ZA,XX,YY) ADDRESS;
    DECLARE (SAVE,I) BYTE,
          X BASED XA BYTE,
          Y BASED YA BYTE,
          Z BASED ZA BYTE;

  /* IF EITHER NUMBER IS ZERO */
    IF (X=0) OR (Y=0) THEN DO;
                Z=0; Z(1)=0; Z(2)=0; RETURN; END;

  /* NUMBERS NON-ZERO */
    XX=SHL(DOUBLE(X),8) OR X(1);
    YY=SHL(DOUBLE(Y),8) OR Y(1);
    ZZ=7FFFH;

    DO I=0 TO 14 ;
          YY= SCR(YY,1);
          IF (CARRY) THEN ZZ=ZZ+XX;
          ZZ=SCR(ZZ,1);
    END;

    ZZ=ZZ+XX;

    IF (CARRY) THEN DO;
          ZZ=SCR(ZZ,1);
          ZE=0;
          END;
          ELSE ZE=-1;

  /* ADD EXPONENTS  */
          SAVE = (X(2) AND 7FH) + (Y(2) AND 7FH) - 40H + ZE;
          IF SAVE > 7FH THEN DO;
          Z=0;    Z(1)=0;    Z(2)=0;
          RETURN;    END;
                Z(2) = SAVE OR ((X(2) XOR Y(2) ) AND 80H);
          Z=HIGH (ZZ);
          Z(1)=LOW (ZZ);
          RETURN;
END MULT;
```

```
   /*                                          */
   /* FLOATING POINT DIVIDE ROUTINE            */
   /*                                          */
DIV: PROCEDURE(XA,YA,ZA);
     DECLARE (XA,YA,ZA,XX,YY,TEMP) ADDRESS,
          X BASED XA BYTE,
          Y BASED YA BYTE,
          Z BASED ZA BYTE,
          (I,SGN) BYTE,
          C(2) BYTE;

     IF X=0H THEN DO;
          Z=0H;  Z(1)=0H;  Z(2)=0H;
          RETURN; END;

     SGN=(X(2) AND 80H) XOR (Y(2) AND 80H);
     XE=X(2) AND 7FH;
     YE=Y(2) AND 7FH;
     ZE=XE-YE+40H;
     XX=SHL(DOUBLE(X),8) OR X(1);
     YY=SHL(DOUBLE(Y),8) OR Y(1);

     IF XX=YY THEN DO;
          ZZ=8000H;  ZE=ZE+1;  GO TO EXIT;
          END;

     IF YY>XX THEN YY=SHR(YY,1);
          ELSE ZE=ZE+1;

     ZZ=0H;
     DO I=1 TO 16;
          TEMP=XX-YY;

          IF CARRY THEN DO;
               IF XX>80H THEN YY=SHR(YY,1);
                    ELSE XX=SHL(XX,1);
               ZZ=SHL(ZZ,1);
               END;
          ELSE DO;
               ZZ=SHL(ZZ,1)+1;
               XX=SHL(TEMP,1);
               END;
          END;
```

```
   /*                                */
   /* OVERFLOW/UNDERFLOW?            */
   /*                                */
EXIT: IF ZE>7FH THEN DO;
         IF XE>YE THEN DO;
               Z=0FFH;  Z(1)=0FFH;  Z(2)=SGN OR 07FH;
               END;
         ELSE DO;
               Z=0;  Z(1)=0;  Z(2)=0;
               END;
         RETURN;
         END;

    Z=HIGH(ZZ);  Z(1)=LOW(ZZ);
    Z(2)=SGN OR ZE;
    RETURN;
END DIV;
```

```
  /*                                                    */
  /* FLOATING POINT COMPARISON ROUTINE                  */
  /*                                                    */
  /*        X < Y  COMPARE=0                            */
  /*        X = Y  COMPARE=1                            */
  /*        X > Y  COMPARE=2                            */

COMPARE: PROCEDURE (XA,YA) BYTE;
    DECLARE (XA,YA,XX,YY) ADDRESS,
          X BASED XA BYTE,
          Y BASED YA BYTE,
          (XSIGN,YSIGN) BYTE;

    XSIGN=X(2) AND 80H;
    YSIGN=Y(2) AND 80H;

  /* SIGNS EQUAL */
    IF (XSIGN=YSIGN) THEN DO;
          XE=X(2) AND 7FH;
          YE=Y(2) AND 7FH;
          XX=SHL(DOUBLE(X),8) OR X(1);
          YY=SHL(DOUBLE(Y),8) OR Y(1);


  /* SIGNS POSITIVE */
          IF XSIGN=0 THEN DO;
                IF XE>YE THEN RETURN 2;
                IF XE<YE THEN RETURN 0;
                IF XX>YY THEN RETURN 2;
                IF XX<YY THEN RETURN 0;
                RETURN 1;
          END;


  /* SIGNS NEGATIVE */
          IF XE>YE THEN RETURN 0;
          IF XE<YE THEN RETURN 2;
          IF XX>YY THEN RETURN 0;
          IF XX<YY THEN RETURN 2;
          RETURN 1;
          END;


  /* SIGNS UNEQUAL */
     IF XSIGN=0 THEN RETURN 2;
     RETURN 0;
END COMPARE;
```

```
  /*                                                  */
  /* FLOATING POINT SQUAREROOT ROUTINE                */
  /*                                                  */
  /* ASSUME VARIABLE IS POSITIVE REAL NUMBER */

SQRT: PROCEDURE (XA,ZA);
    DECLARE (XA,ZA) ADDRESS,
          X BASED XA BYTE,
          Z BASED ZA BYTE,
          R BYTE,
          E(3) BYTE,
          T(3) BYTE;

    ZE=X(2)-40H;

  /* INITIAL APPROXIAMATION OF ROOT IS */
  /*       (1+MANT)/2 * EXP/2          */

    IF ZE<80H THEN T(2)=SHR(ZE,1)+40H;
    ELSE T(2)=40H-SHR(-ZE,1);

    ZZ=SHL(DOUBLE(X),8) OR X(1);
    ZZ=SHR(ZZ,1) OR 8000H;
    T=HIGH(ZZ);  T(1)=LOW(ZZ);

    DO R=1 TO 3;
          CALL DIV (XA,.T,.B);
          CALL ADD (.B,.T,.T);
          T(2)=T(2)-1;
          END;

    Z=T;  Z(1)=T(1);
    Z(2)=T(2);

    RETURN;
END SQRT;
```

```
   /*                                        */
   /* FLOATING POINT COSINE FUNCTION         */
   /*                                        */
   /*     0.0 <= THETA <= PI/2               */
   /*                                        */
COS$FUNC: PROCEDURE (THA,MAGA);
     DECLARE (THA,MAGA) ADDRESS,
          N BYTE,
          THETA BASED THA BYTE,
          DIF(3) BYTE,
          TEMP(3) BYTE,

          CRD DATA (80H,00H,3DH,0C0H,00H,3EH,0ACH,00H,3FH,
          0E0H,00H,3FH,90H,00H,40H,0B0H,00H,40H,0DCH,00H,40H,
          CF0H,00H,40H,88H,00H,41H,98H,00H,41H,0A8H,00H,41H,
          0B8H,00H,41H,0C8H,00H,41H),

          COS DATA (0FFH,80H,40H,0FBH,83H,40H,0F3H,9AH,40H,
          0E7H,0E3H,40H,0D8H,8FH,40H,0C5H,0D9H,40H,0B0H,0CH,40H,
          97H,81H,40H,0F9H,2FH,3FH,0BFH,7AH,3FH,82H,CC8H,3FH,
          88H,17H,3EH,87H,0EDH,3AH),

          SIN DATA (0FFH,0D5H,3CH,0BEH,0E1H,3EH,9DH,69H,3FH,
          0D8H,0ECH,3FH,88H,87H,40H,0A2H,76H,40H,0B9H,0DCH,40H,
          0CEH,5BH,40H,0DFH,0A3H,40H,0ECH,6DH,40H,0F7H,82H,40H,
          0FDH,0BBH,40H,0FFH,0FEH,40H);

     IF THETA(2)>3DH THEN DO;
          ZE=8 - (THETA(2)-3DH);
          N=3 * SHR(THETA,ZE);
          END;
     ELSE  N=0;

     CALL SUB (THA,.ORD(N),.DIF);
     CALL MULT (.DIF,.COS(N),.TEMP);
     TEMP(2)=TEMP(2) - 1;
     CALL ADD (.TEMP,.SIN(N),.TEMP);
     CALL MULT (.TEMP,.DIF,.TEMP);
     CALL SUB (.COS(N),.TEMP,MAGA);
END COS$FUNC;
```

```
   /*                                  */
   /* FLOATING POINT ATAN FUNCTION     */
   /*          0.0 <= X < 1.0          */
   /*                                  */
ATAN$FUNC: PROCEDURE (XA,THA);
    DECLARE (XA,THA) ADDRESS,
          N BYTE,
          X BASED XA BYTE,
          DIF(3) BYTE,
          TEMP(3) BYTE,

          ORD DATA (80H,00H,0BCH,0CCH,00H,0BDH,0A0H,00H,0BEH,
          0E0H,00H,0BEH,90H,00H,0BFH,0BCH,00H,0BFH,0C0H,00H,0BFH,
          0FCH,00H,0BFH,88H,00H,0C0H,98H,00H,0C0H,0A8H,00H,0C0H,
          0B8H,00H,0C0H,0C8H,00H,0C0H,0D8H,00H,0C0H,0E8H,00H,0C0H,
          0F8H,00H,0C0H,84H,20H,0C1H),

          VAL DATA (0FFH,0EBH,3BH,0BFH,71H,3DH,9EH,0B7H,3EH,
          0DCH,87H,3EH,8CH,60H,3FH,0A9H,85H,3FH,0C5H,92H,3FH,
          0E0H,6EH,3FH,0FAH,07H,3FH,89H,2BH,40H,94H,0ACH,40H,
          9FH,8AH,40H,0A9H,0C8H,40H,0B3H,6BH,40H,0BCH,7BH,40H,
          0C5H,0CH,40H,0CDH,00H,40H),

          SLOPE DATA (0FFH,0C0H,40H,0FDH,0C5H,40H,0F9H,0E6H,40H,
          0F4H,4FH,40H,0EDH,3CH,40H,0E4H,0F2H,40H,0DBH,0BCH,40H,
          0D1H,0E2H,40H,0C7H,0A7H,40H,0BCH,46H,40H,0B2H,0F0H,40H,
          0A8H,0CCH,40H,9EH,0F9H,40H,95H,08AH,40H,8CH,8FH,40H,
          84H,10H,40H,0F8H,20H,3FH),

          RATE DATA (0FFH,80H,0BBH,0BCH,0ABH,0BDH,98H,77H,0BEH,
          0CCH,02H,0BEH,0F7H,53H,0BEH,8CH,0C4H,0BFH,99H,3EH,0BFH,
          0A1H,52H,0BFH,0A5H,71H,0BFH,0A6H,2EH,0BFH,0A4H,29H,0BFH,
          9FH,0FFH,0BFH,9AH,3FH,0BFH,93H,68H,0BFH,8BH,0E2H,0BFH,
          83H,0FFH,0BFH,0F8H,02H,0BEH);

    IF X(2) > 3CH THEN DO;
          ZE=8-(X(2) - 3CH);
          N=3*SHR(X,ZE);
          END;
    ELSE N=0;

    CALL ADD (XA, .ORD(N), .DIF);
    CALL MULT (.DIF, .RATE(N), .TEMP);
    CALL ADD (.TEMP, .SLOPE(N), .TEMP);
    CALL MULT (.TEMP, .DIF, .TEMP);
    CALL ADD (.VAL(N), .TEMP, THA);

END ATAN$FUNC;
```

```
  /*                                                    */
  /* FLOATING POINT TRIGONOMETRY ROUTINE                */
  /*                                                    */
TRIG: PROCEDURE (XA,YA,THA,N);
  /*     N=1    COS          */
  /*     N=2    SIN          */
  /*     N=3   COS & SIN     */
  /*     N=4    ATAN         */

    DECLARE (XA,YA,THA) ADDRESS,
          (N,I) BYTE,
          X BASED XA BYTE,
          Y BASED YA BYTE,
          TH BASED THA BYTE,
          TEMP(3) BYTE,
          THETA(3) BYTE,
          TEMPX (3) BYTE,
          PIHALF  DATA (0C9H,10H,41H),
          PI      DATA (0C9H,10H,42H),
          PIONEHF DATA (96H,0CCH,43H),
          PITWO DATA (0C9H,10H,43H),
          MPIHALF DATA (0C9H,10H,0C1H);


    IF N<4 THEN DO;
  /* SIN AND COS */
          THETA=TH;  THETA(1)=TH(1);  THETA(2)=TH(2);
          IF THETA=0 THEN GO TO EXIT;

          DO WHILE THETA(2)>80H;
               CALL ADD (.THETA,.PITWO,.THETA);
               END;

          DO WHILE THETA(2) > 43H;
               CALL SUB (.THETA,.PITWO,.THETA);
               END;

          DO CASE COMPARE (.THETA,.PITWO);
               ;    /* THETA LESS THAN TWO PI */
                 /* THETA EQUALS TWO PI */
               GO TO EXIT;
                 /* THETA GREATER THAN TWO PI */
               CALL SUB (.THETA,.PITWO,.THETA);
               END;
```

```
   I=0;
   DO WHILE THETA(2) < 80H;
         TEMP=THETA;  TEMP(1)=THETA(1);  TEMP(2)=THETA(2);
         I=I+1;
         CALL ADD (.THETA,.MPIHALF,.THETA);
         END;

   THETA(2)=THETA(2) AND 7FH;

/* COS */
   IF N THEN DO CASE I-1;

         CALL COS$FUNC (.TEMP,XA);

         DO;
               CALL COS$FUNC (.THETA,XA);
               X(2)=X(2) OR 80H;
               END;

         DO;
               CALL COS$FUNC (.TEMP,XA);
               X(2)=X(2) OR 80H;
               END;

         CALL COS$FUNC(.THETA,XA);
         END;

/* SIN */
   IF SHR (N,1) THEN DO CASE I-1;

         CALL COS$FUNC (.THETA,YA);

         CALL COS$FUNC (.TEMP,YA);

         DO;
               CALL COS$FUNC (.THETA,YA);
               Y(2)=Y(2) OR 80H;
               END;

         DO;
               CALL COS$FUNC (.TEMP,YA);
               Y(2)=Y(2) OR 80H;
               END;

         END;
```

```
            RETURN;

EXIT:
            X=80H;   X(1)=00H;   X(2)=41H;
            Y=00H;   Y(1)=00H;   Y(2)=00H;
            RETURN;
            END;



   /* ATAN */
     TEMPX=X;  TEMPX(1)=X(1);  TEMPX(2)=X(2) AND 7FH;
     TEMP=Y;   TEMP(1)=Y(1);   TEMP(2)=Y(2) AND 7FH;
     I=COMPARE(.TEMPX,.TEMP);
     DO CASE I;

          /* X < Y */
          DO;
               CALL DIV (.TEMPX,.TEMP,.TEMP);
               CALL ATAN$FUNC (.TEMP,THA);
               END;

          /* X = Y */
          DO;
               TH=0C9H;
               TH(1)=10H;
               TH(2)=40H;
               END;

          /* X > Y */
          DO;
               CALL DIV (.TEMP,.TEMPX,.TEMP);
               CALL ATAN$FUNC (.TEMP,THA);
               END;
          END;

     IF X(2) < 80H THEN DO;

          IF Y(2) < 80H THEN DO;
               /* QUADRANT I */
               IF I=0 THEN CALL SUB (.PIHALF,THA,THA);
               RETURN;  END;

          ELSE DO;
               /* QUADRANT IV */
               IF I=2 THEN CALL SUB (.PITWO,THA,THA);
```

```
                ELSE CALL ADD (.PIONEHF,THA,THA);
                RETURN;  END;
          END;

     IF Y(2) < 80H THEN DO;
          /* QUADRANT II */
          IF I=2 THEN CALL SUB (.PI,THA,THA);
          ELSE CALL ADD (.PIHALF,THA,THA);
          RETURN;  END;

     ELSE DO;
          /* QUADRANT III */
          IF I=0 THEN CALL SUB (.PIONEHF,THA,THA);
          ELSE CALL ADD (.PI,THA,THA);
          RETURN;  END;

END TRIG;
```

```
MON1: PROCEDURE (FUNC,INFO);
    DECLARE FUNC BYTE, INFO ADDRESS;
    GO TO 3FFDH;
    END MON1;


CRLF: PROCEDURE;
  /* SEND CARRIAGE-RETURN-LINE-FEED */
    CALL MON1 (2,0DH);
    CALL MON1 (2,0AH);
    END CRLF;


PRINT: PROCEDURE (A);
  /* PRINT THE BUFFER STARTING AT A */
    DECLARE A ADDRESS;
    CALL MON1 (9,A);
    END PRINT;


  /*                                                    */
  /* PRINT A FLOATING POINT NUMBER WITH SIGN            */
  /* AND DECIMAL POINT                                  */
  /*      XA=ADDRESS OF VARIABLE                        */
  /*      PT=NR OF DIGITS TO THE RT OF THE              */
  /*         DECIMAL POINT                              */
  /* NOTE: INPUT VARIABLE MUST BE LESS THAN             */
  /*            65536.                                  */
  /* FIELD WILL BE 7 SPACES WIDE                        */
  /*          0 <= PT <= 5                              */
  /*                                                    */

PRINT$FPT$DEC: PROCEDURE (XA,PT);
    DECLARE (XA,DEC) ADDRESS,
          X BASED XA BYTE,
          Y(3) BYTE,
          HALF DATA (80H,00H,40H),
          BUFFER (8) BYTE,
          (RADIX,I,N,ZSUP,SYMBOL,PT)BYTE,

          RATIO DATA (0A0H,00H,44H,0C8H,00H,47H,
          0FAH,00H,4AH,9CH,40H,4EH,0C3H,50H,51H);

    IF ((X(2) AND 7FH)>50H) THEN GO TO STAR;

    IF PT>0 THEN DO;
          N=3*(PT-1);
```

```
      CALL MULT (XA,.RATIC(N),.Y);
      END;
ELSE DO;
      Y=X; Y(1)=X(1);  Y(2)=X(2);
      END;

IF Y(2)<80H THEN CALL ADD (.Y,.HALF,.Y);
ELSE CALL SUB (.Y,.HALF,.Y);

YE=Y(2) AND 7FH;
IF YE > 50H THEN GO TO STAR;

RADIX=6-PT;
BUFFER=' ';
BUFFER(RADIX)='.';
SYMBOL=0;
ZZ=SHL(DOUBLE(Y),8) OR Y(1);

IF YE<50H THEN DO;
      YE=50H - YE;
      ZZ=SHR(ZZ,YE);
      END;

IF RADIX=1 THEN DO;
      ZSUP=0;
      N=2;
      END;
ELSE DO;
      ZSUP=1;
      N=1;
      END;

IF ZZ=0 THEN DO;
      DO WHILE (N+1)<RADIX;
            BUFFER(N)=' ';
            N=N+1;
            END;
      GO TO ZEROS;
      END;

DEC=2710H;
DO I=1 TO 5;
```

```
            IF N=RADIX THEN DO;
                  N=N+1;
                  ZSUP=0;
                  END;
            BUFFER(N)=ZZ/DEC + '0';
            ZZ=ZZ MOD DEC;
            DEC=DEC/10;

            IF ZSUP THEN DO;
                  IF BUFFER(N)='0' THEN DO;
                        SYMBOL=N;
                        BUFFER(N)=' ';
                        END;
                  ELSE ZSUP=0;
            END;

            N=N + 1;
      END;

      IF (X(2)>=80H) THEN BUFFER(SYMBOL)='-';

ZEROS: DO WHILE N<7;
            IF N=RADIX THEN N = N + 1;
            BUFFER(N)='0';
            N=N + 1;
      END;

EXIT: BUFFER(7)='$';
      CALL PRINT (.BUFFER);
      RETURN;

STAR: DO N=0 TO 6;
                  BUFFER(N)='*';
                  END;
            GO TO EXIT;
END PRINT$FPT$DEC;
```

```
     DECLARE (XEST,YEST,VXEST,VYEST) ADDRESS,
          (P11,P12,P13,P14,P21,P22,P23,P24) ADDRESS,
          (P31,P32,P33,P34,P41,P42,P43,P44) ADDRESS,
          (ITER,CONTROL,MARKOLD,DFLAG,K,N,N1,N2) BYTE,
          (TPA,TPB) BYTE,
          (DELX,DELY) (3) BYTE,
          (SIGX,SIGY,SIGXY,ACCUM) (3) BYTE,
          (GMT1,GMT2,GMT3,GMT4) (3) BYTE,
          (GMT5,GMT6,GMT7,GMT8) (3) BYTE,
          (MRNG,MBRG,MTIM,T,TSC) (3) BYTE,
          (TEMPCOS,TEMPSIN,PTA,PP1,PP2) (3) BYTE,
          (AXX,AXY,AYY,AYX) (3) BYTE,
          (BXXT,BXYT,BYYT,BYXT) (3) BYTE,
          (PT01,PT02,PT03) (3) BYTE,
          (PT11,PT12,PT13,PT14) (3) BYTE,
          (PT21,PT22,PT23,PT24) (3) BYTE,
          (PT31,PT32,PT33,PT34) (3) BYTE,
          (PT41,PT42,PT43,PT44) (3) BYTE,
          (PX11,PX12,PX13,PX14) (24) BYTE,
          (PX21,PX22,PX23,PX24) (24) BYTE,
          (PX31,PX32,PX33,PX34) (24) BYTE,
          (PX41,PX42,PX43,PX44) (24) BYTE,
          (FIL1,FIL2,FIL3,FIL4,FIL5) (24) BYTE,

SIMFLAG DATA (02H,08H,20H,04H,10H,40H,80H,01H,
              ------------
              02H,08H,20H,04H,80H),

BRG DATA (0A3H,086H,042H,
     0AFH,097H,043H,
     0AFH,0E5H,043H,
      --------
      0AFH,068H,043H),
```

```
RNG DATA (0DFH,080H,048H,
      0E4H,0B0H,04DH,
      0F9H,060H,04EH,

      - - - - - - - - - -

       0E2H,07EH,050H),

TIM DATA (088H,013H,041H,
      0C5H,032H,042H,
      0C9H,080H,042H,

      - - - - - - - - - -

      0CEH,09CH,043H),


NSVEL DATA (0E4H,29H,44H),
EWVEL DATA (0CH,00H,00H),
DEG$RAD DATA (0E5H,2FH,46H),
KN$YS DATA (0E3H,58H,41H),
ONE DATA (80H,00H,41H),
EIGHT DATA (80H,00H,44H),
SIGA DATA (83H,12H,37H),
SIGTH DATA (0A3H,69H,2FH),
SIGR DATA (0CCH,8FH,46H);
DECLARE (ACC1,ACC2) (3) BYTE;



MARKCLC=0;
CONTRCL=0FFH;
ITER=0;

DO WHILE ITER < 85;

     /* CHECK CONTROL PANEL */
     MARKCLC=MARKCLC AND CONTRCL;

     /*                    */
     /* CHECK FOR NEW DATA */
     /*                    */
     /* INPUT CFLAG */
     CFLAG=SIMFLAG(ITER);
```

```
   K=0;
   TPA=1;

   DO WHILE NOT DFLAG;
         DFLAG=SHR(DFLAG,1);
         TPA=SHL(TPA,1);
         K=K+1;
         END;


/*                                                    */
/* INPUT DATA FOR INTERFACE UNIT K                    */
/*                                                    */
   N=3*ITER;    N1=N+1;    N2=N+2;
   MRNG=RNG(N);    MRNG(1)=RNG(N1);    MRNG(2)=RNG(N2);
   MBRG=BRG(N);    MBRG(1)=BRG(N1);    MBRG(2)=BRG(N2);
   MTIM=TIM(N);    MTIM(1)=TIM(N1);    MTIM(2)=TIM(N2);


   ITER=ITER+1;
   DFLAG=DFLAG AND NOT TPA;
/* END INTERFACE SIMULATOR */
```

```
IF K=0 THEN TPB=MARKOLD;
     ELSE TPB=SHR(MARKOLD,K);

N=3*K;    N1=N+1;    N2=N+2;


IF NOT TPB THEN DO;
     MARKOLD=MARKOLD OR TPA;

     PX11(N)=0BEH;    PX11(N1)=3CH;    PX11(N2)=4EH;
     PX12(N)=0;    PX12(N1)=0;    PX12(N2)=0;
     PX13(N)=0;    PX13(N1)=0;    PX13(N2)=0;
     PX14(N)=0;    PX14(N1)=0;    PX14(N2)=0;
     PX21(N)=0;    PX21(N1)=0;    PX21(N2)=0;
     PX22(N)=0D0H;    PX22(N1)=40H;    PX22(N2)=4AH;
     PX23(N)=0;    PX23(N1)=0;    PX23(N2)=0;
     PX24(N)=0;    PX24(N1)=0;    PX24(N2)=0;
     PX31(N)=0;    PX31(N1)=0;    PX31(N2)=0;
     PX32(N)=0;    PX32(N1)=0;    PX32(N2)=0;
     PX33(N)=0BEH;    PX33(N1)=3CH;    PX33(N2)=4EH;
     PX34(N)=0;    PX34(N1)=0;    PX34(N2)=0;
     PX41(N)=0;    PX41(N1)=0;    PX41(N2)=0;
     PX42(N)=0;    PX42(N1)=0;    PX42(N2)=0;
     PX43(N)=0;    PX43(N1)=0;    PX43(N2)=0;
     PX44(N)=0D0H;    PX44(N1)=40H;    PX44(N2)=4AH;
     CALL TRIG (.TEMPCOS,.TEMPSIN,.MBRG,3);
     CALL MULT(.TEMPSIN,.MRNG,.FIL1(N));
     CALL MULT(.TEMPCOS,.MRNG,.FIL3(N));
     FIL2(N)=0;    FIL2(N1)=0;    FIL2(N2)=0;
     FIL4(N)=0;    FIL4(N1)=0;    FIL4(N2)=0;
     FIL5(N)=MTIM;    FIL5(N1)=MTIM(1);    FIL5(N2)=MTIM(2);
     END;


ELSE DO;

     CALL SUB(.MTIM,.FIL5(N),.T);

     FIL5(N)=MTIM;    FIL5(N1)=MTIM(1);    FIL5(N2)=MTIM(2);

     IF T(2)>7FH THEN CALL ADD (.T,.EIGHT,.T);


     P11=.PX11(N);
     P12=.PX12(N);
     P13=.PX13(N);
     P14=.PX14(N);
```

```
P21=.PX21(N);
P22=.PX22(N);
P23=.PX23(N);
P24=.PX24(N);
P31=.PX31(N);
P32=.PX32(N);
P33=.PX33(N);
P34=.PX34(N);
P41=.PX41(N);
P42=.PX42(N);
P43=.PX43(N);
P44=.PX44(N);


CALL MULT(.T,.T,.TSQ);
CALL MULT(.SIGA,.TSQ,.PT01);

CALL MULT(.PT01,.TSQ,.PT02);
PT02(2)=PT02(2)-2;

CALL MULT(.PT01,.T,.PT03);
PT03(2)=PT03(2)-1;


CALL ADD (P12,P21,.PT11);
CALL MULT(.PT11,.T,.PT11);
CALL ADD(.PT11,P11,.PT11);
CALL ADD(.PT11,.PT02,.PT11);
CALL MULT(P22,.TSQ,.ACCUM);
CALL ADD(.PT11,.ACCUM,.PT11);

CALL MULT(P22,.T,.PTA);
CALL ADD(.PTA,.PT03,.PTA);

CALL ADD(P12,.PTA,.PT12);

CALL ADD(P21,.PTA,.PT21);

CALL ADD(P14,P23,.PT13);
CALL MULT(.PT13,.T,.PT13);
CALL ADD(.PT13,P13,.PT13);
CALL MULT(P24,.TSQ,.ACCUM);
CALL ADD(.PT13,.ACCUM,.PT13);

CALL MULT(P24,.T,.PTA);
```

```
CALL ADD(P14,.PTA,.PT14);

CALL ADD(P23,.PTA,.PT23);

CALL ADD(P22,.PTO1,.PT22);

CALL ADD(P32,P41,.PT31);
CALL MULT(.PT31,.T,.PT31);
CALL ADD(P31,.PT31,.PT31);
CALL MULT(P42,.TSQ,.ACCUM);
CALL ADD(.PT31,.ACCUM,.PT31);

CALL MULT(P42,.T,.PTA);

CALL ADD(P32,.PTA,.PT32);

CAL_ ADD(P41,.PTA,.PT41);

CALL ADD(P34,P43,.PT33);
CALL MULT(.PT33,.T,.PT33);
CALL ADD(P33,.PT33,.PT33);
CALL ADD(.PT33,.PTO2,.PT33);
CALL MULT(P44,.TSQ,.ACCUM);
CALL ADD(.PT33,.ACCUM,.PT33);

CALL MULT(P44,.T,.PTA);
CALL ADD(.PTA,.PTO3,.PTA);

CALL ADD(P34,.PTA,.PT34);

CALL ADD(P43,.PTA,.PT43);

CALL ADD(P44,.PTO1,.PT44);


CALL TRIG(.TEMPCOS,.TEMPSIN,.MRNG,3);
CALL MULT(.TEMPCOS,.TEMPCOS,.GMT1);

CALL MULT(.TEMPSIN,.TEMPSIN,.GMT2);

CALL MULT(.MRNG,.MRNG,.GMT3);
CALL MULT(.GMT3,.SIGTH,.GMT3);


CALL MULT(.SIGR,.GMT1,.SIGY);
CALL MULT(.GMT3,.GMT2,.ACCUM);
CALL ADD(.SIGY,.ACCUM,.SIGY);
```

```
CALL MULT(.SIGR,.GMT2,.SIGX);
CALL MULT(.GMT3,.GMT1,.ACCUM);
CALL ADD(.SIGX,.ACCUM,.SIGX);

CALL MULT (.TEMPCOS,.TEMPSIN,.SIGXY);
CALL SUB(.SIGR,.GMT3,.ACCUM);
CALL MULT(.SIGXY,.ACCUM,.SIGXY);


CALL ADD(.PT11,.SIGX,.GMT4);

CALL ADD(.PT13,.SIGXY,.GMT5);

CALL ADD(.PT31,.SIGXY,.GMT6);

CALL ADD(.PT33,.SIGY,.GMT7);

CALL MULT(.GMT4,.GMT7,.GMT8);
CALL MULT(.GMT5,.GMT6,.ACCUM);
CALL ASUB(.GMT8,.ACCUM,.GMT8);


CALL MULT(.PT11,.GMT7,.AXX);
CALL MULT(.PT13,.GMT6,.ACCUM);
CALL SUB(.AXX,.ACCUM,.AXX);
CALL DIV(.AXX,.GMT8,.AXX);

CALL MULT(.PT13,.GMT4,.AXY);
CALL MULT(.PT11,.GMT5,.ACCUM);
CALL SUB(.AXY,.ACCUM,.AXY);
CALL DIV(.AXY,.GMT8,.AXY);

CALL MULT(.PT21,.GMT7,.BXXT);
CALL MULT(.PT23,.GMT6,.ACCUM);
CALL SUB(.BXXT,.ACCUM,.BXXT);
CALL DIV(.BXXT,.GMT8,.BXXT);

CALL MULT(.PT23,.GMT4,.BXYT);
CALL MULT(.PT21,.GMT5,.ACCUM);
CALL SUB(.BXYT,.ACCUM,.BXYT);
CALL DIV(.BXYT,.GMT8,.BXYT);

CALL MULT(.PT31,.GMT7,.AYX);
CALL MULT(.PT33,.GMT6,.ACCUM);
CALL SUB(.AYX,.ACCUM,.AYX);
CALL DIV(.AYX,.GMT8,.AYX);
```

```
CALL MULT(.PT33,.GMT4,.AYY);
CALL MULT(.PT31,.GMT5,.ACCUM);
CALL SUB(.AYY,.ACCUM,.AYY);
CALL DIV(.AYY,.GMT8,.AYY);

CALL MULT(.PT41,.GMT7,.BYXT);
CALL MULT(.PT43,.GMT6,.ACCUM);
CALL SUB(.BYXT,.ACCUM,.BYXT);
CALL DIV(.BYXT,.GMT8,.BYXT);

CALL MULT(.PT43,.GMT4,.BYYT);
CALL MULT(.PT41,.GMT5,.ACCUM);
CALL SUB(.BYYT,.ACCUM,.BYYT);
CALL DIV(.BYYT,.GMT8,.BYYT);


CALL SUB(.ONE,.AXX,.PP1);

CALL SUB(.ONE,.AYY,.PP2);


CALL MULT(.PP1,.PT11,.ACCUM);
CALL MULT(.AXY,.PT31,P11);
CALL SUB(.ACCUM,P11,P11);

CALL MULT(.PP1,.PT12,.ACCUM);
CALL MULT(.AXY,.PT32,P12);
CALL SUB(.ACCUM,P12,P12);

CALL MULT(.PP1,.PT13,.ACCUM);
CALL MULT(.AXY,.PT33,P13);
CALL SUB(.ACCUM,P13,P13);

CALL MULT(.PP1,.PT14,.ACCUM);
CALL MULT(.AXY,.PT34,P14);
CALL SUB(.ACCUM,P14,P14);

CALL MULT(.BXXT,.PT11,P21);
CALL MULT(.BXYT,.PT31,.ACCUM);
CALL ADD(P21,.ACCUM,P21);
CALL SUB(.PT21,P21,P21);

CALL MULT(.BXXT,.PT12,P22);
CALL MULT(.BXYT,.PT32,.ACCUM);
CALL ADD(P22,.ACCUM,P22);
CALL SUB(.PT22,P22,P22);
```

```
CALL MULT(.BXXT,.PT13,P23);
CALL MULT(.BXYT,.PT33,.ACCUM);
CALL ADD(P23,.ACCUM,P23);
CALL SUB(.PT23,P23,P23);

CALL MULT(.BXXT,.PT14,.ACCUM);
CALL SUB(P24,.ACCUM,P24);
CALL MULT(.BXYT,.PT34,.ACCUM);
CALL SUB(P24,.ACCUM,P24);

CALL MULT(.PP2,.PT31,P31);
CALL MULT(.AYX,.PT11,.ACCUM);
CALL SUB(P31,.ACCUM,P31);

CALL MULT(.PP2,.PT32,P32);
CALL MULT(.AYX,.PT12,.ACCUM);
CALL SUB(P32,.ACCUM,P32);

CALL MULT(.PP2,.PT33,P33);
CALL MULT(.AYX,.PT13,.ACCUM);
CALL SUB(P33,.ACCUM,P33);

CALL MULT(.PP2,.PT34,P34);
CALL MULT(.AYX,.PT14,.ACCUM);
CALL SUB(P34,.ACCUM,P34);

CALL MULT(.BYXT,.PT11,P41);
CALL MULT(.BYYT,.PT31,.ACCUM);
CALL ADD(P41,.ACCUM,P41);
CALL SUB(.PT41,P41,P41);

CALL MULT(.BYXT,.PT12,.ACCUM);
CALL SUB(P42,.ACCUM,P42);
CALL MULT(.BYYT,.PT32,.ACCUM);
CALL SUB(P42,.ACCUM,P42);

CALL MULT(.BYXT,.PT13,P43);
CALL MULT(.BYYT,.PT33,.ACCUM);
CALL ADD(P43,.ACCUM,P43);
CALL SUB(.PT43,P43,P43);

CALL MULT(.BYXT,.PT14,P44);
CALL MULT(.BYYT,.PT34,.ACCUM);
CALL ADD(P44,.ACCUM,P44);
CALL SUB(.PT44,P44,P44);
```

```
XEST=.FIL1(N);
VXEST=.FIL2(N);
YEST=.FIL3(N);
VYEST=.FIL4(N);

CALL MULT(VXEST,.T,.ACCUM);
CALL ADD(.ACCUM,XEST,XEST);

CALL MULT(VYEST,.T,.ACCUM);
CALL ADD(.ACCUM,YEST,YEST);


CALL MULT(.MRNG,.TEMPSIN,.ACCUM);
CALL SUB(.ACCUM,XEST,.DELX);

CALL MULT(.MRNG,.TEMPCOS,.ACCUM);
CALL SUB(.ACCUM,YEST,.DELY);


CALL MULT(.AXX,.DELX,.ACCUM);
CALL ADD(XEST,.ACCUM,XEST);
CALL MULT(.AXY,.DELY,.ACCUM);
CALL ADD(XEST,.ACCUM,XEST);

CALL MULT(.BXXT,.DELX,.ACCUM);
CALL ADD(VXEST,.ACCUM,VXEST);
CALL MULT(.BXYT,.DELY,.ACCUM);
CALL ADD(VXEST,.ACCUM,VXEST);

CALL MULT(.AYY,.DELY,.ACCUM);
CALL ADD(YEST,.ACCUM,YEST);
CALL MULT(.AYX,.DELX,.ACCUM);
CALL ADD(YEST,.ACCUM,YEST);

CALL MULT(.BYYT,.DELY,.ACCUM);
CALL ADD(VYEST,.ACCUM,VYEST);
CALL MULT(.BYXT,.DELX,.ACCUM);
CALL ADD(VYEST,.ACCUM,VYEST);


END;
```

```
   CALL CRLF;
   CALL CRLF;
   CALL MON1 (2,K+'1');
   CALL PRINT (.'     $');
   CALL PRINT$FPT$DEC (.MTIM,3);
   CALL CRLF;
   CALL TRIG (.FIL3(N),.FIL1(N),.ACCUM,4);
   CALL MULT (.ACCUM,.DEG$RAD,.ACCUM);
   CALL CRLF;
   CALL PRINT (.'BEARING = $');
   CALL PRINT$FPT$DEC (.ACCUM,2);

   CALL MULT (.FIL1(N),.FIL1(N),.ACC1);
   CALL MULT (.FIL3(N),.FIL3(N),.ACC2);
   CALL ADD (.ACC1,.ACC2,.ACC1);
   CALL SQRT (.ACC1,.ACCUM);
   CALL CRLF;
   CALL PRINT (.'RANGE = $');
   CALL PRINT$FPT$DEC (.ACCUM,0);

   CALL ADD (.FIL2(N),.EWVEL,.ACC1);
   CALL ADD (.FIL4(N),.NSVEL,.ACC2);
   CALL TRIG (.ACC2,.ACC1,.ACCUM,4);
   CALL MULT (.ACCUM,.DEG$RAD,.ACCUM);
   CALL CRLF;
   CALL PRINT (.'COURSE = $');
   CALL PRINT$FPT$DEC (.ACCUM,2);

   CALL MULT (.ACC1,.ACC1,.ACCUM);
   CALL MULT (.ACC2,.ACC2,.ACC1);
   CALL ADD (.ACCUM,.ACC1,.ACCUM);
   CALL SQRT (.ACCUM,.ACC1);
   CALL MULT (.ACC1,.KN$YS,.ACCUM);
   CALL CRLF;
   CALL PRINT (.'SPEED = $');
   CALL PRINT$FPT$DEC (.ACCUM,3);

   CALL CRLF;
   CALL CRLF;

       END;

   EOF
```

LIST OF REFERENCES

1. INTEL Corporation, INTEL 8080 Microcomputer Systems User's Manual, September 1975.

2. "Advanced Microprocessors," Electronics, v. 49, p. 10-11, 1 April 1976.

3. Castella, F. R., and Dunnebacke, F. G., Analytical Results for the X, Y Kalman Tracking Filter, IEEE Transactions on Aerospace and Electronic Systems, November 74.

4. Gelb, Arthur, (ed.), Applied Optimal Estimation, M.I.T. Press, 1974.

5. Skolnik, M. I., Introduction to Radar Systems, McGraw-Hill, 1962.

6. NAVSHIPS 93684, Technical Manual for Radar Set AN/SPS-10D, v. 1.

7. NAVSHIPS 91921,32 (A), Performance Standard Sheet for Radar Set AN/SPS-10 (Series), 17 April 1963.

8. NAVSHIPS 94120, Complimentary Technical Manual for AN/SPS-10E, 13 June 1961.

9. "Microprocessors," Electronics, v. 49, 15 April, 1976.

10. Flynn, G., Transistor-Transistor Logic, Howard W. Sams and Co., Inc., 1973.

11. Demetry, J. S., Notes on the Theory and Applications of Optimal Estimation, U. S. Naval Postgraduate School, Monterey, California, p. 93, 1970.

12. Flores, I., The Logic of Computer Arithmetic, Prentice-Hall, Inc., 1963.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor D. E. Kirk, Code 52 Ki Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Asst Professor V. M. Powers, Code 52 Pw Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	3
5.	LT Charles Howard Wilson, USN 7401 Eastmoreland Road, Apt. 514 Annandale, Virginia 22003	1